USGS
science for a changing world

User Guide to Exploration and Graphics for RivEr Trends (EGRET) and dataRetrieval: R Packages for Hydrologic Data

Chapter 10 of
Section A, Statistical Analysis
Book 4, Hydrologic Analysis and Interpretation

Techniques and Methods 4–A10
Version 2.0, February 2015

U.S. Department of the Interior
U.S. Geological Survey

User Guide to Exploration and Graphics for RivEr Trends (EGRET) and dataRetrieval: R Packages for Hydrologic Data

By Robert M. Hirsch and Laura A. De Cicco

Chapter 10 of
Section A, Statistical Analysis
Book 4, Hydrologic Analysis and Interpretation

Techniques and Methods 4–A10
Version 2.0, February 2015

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
SALLY JEWELL, Secretary

U.S. Geological Survey
Suzette M. Kimball, Acting Director

U.S. Geological Survey, Reston, Virginia:
First release: 2014
Revised: February 2015 (ver. 2.0)

Suggested citation:
Hirsch, R.M., and De Cicco, L.A., 2015, User guide to Exploration and Graphics for RivEr Trends (EGRET) and dataRetrieval—R packages for hydrologic data (version 2.0, February 2015): U.S. Geological Survey Techniques and Methods book 4, chap. A10, 93 p., *http://dx.doi.org/10.3133/tm4A10*.

ISSN 2328-7055 (online)

Contents

Figures

Tables

Conversion Factors

Inch/pound to International System of Units

Multiply	By	To obtain
	Flow rate	
cubic foot per second (ft^3/s; cfs)	0.02832	cubic meter per second (m^3/s; cms)

International System of Units to Inch/pound

Multiply	By	To obtain
	Length	
millimeter (mm)	0.03937	inch (in.)
meter (m)	3.281	foot (ft)
kilometer (km)	0.6214	mile (mi)
	Area	
square kilometer (km^2)	0.3861	square mile (mi^2)
	Volume	
liter (L)	33.82	ounce, fluid (fl. oz)
liter (L)	2.113	pint (pt)
liter (L)	1.057	quart (qt)
liter (L)	0.2642	gallon (gal)
cubic meter (m^3)	264.2	gallon (gal)
cubic meter (m^3)	0.0002642	million gallons (Mgal)
cubic meter (m^3)	35.31	cubic foot (ft^3)
cubic meter (m^3)	1.308	cubic yard (yd^3)
cubic meter (m^3)	0.0008107	acre-foot (acre-ft)
	Flow rate	
cubic meter per second (m^3/s; cms)	70.07	acre-foot per day (acre-ft/d)
cubic meter per second (m^3/s; cms)	35.31	cubic foot per second (ft^3/s; cfs)
	Mass	
kilogram (kg)	2.205	pound avoirdupois (lb)

User Guide to Exploration and Graphics for RivEr Trends (EGRET) and dataRetrieval: R Packages for Hydrologic Data

By Robert M. Hirsch and Laura A. De Cicco

Abstract

Evaluating long-term changes in river conditions (water quality and discharge) is an important use of hydrologic data. To carry out such evaluations, the hydrologist needs tools to facilitate several key steps in the process: acquiring the data records from a variety of sources, structuring it in ways that facilitate the analysis, processing the data with routines that extract information about changes that may be happening, and displaying findings with graphical techniques. An R package called EGRET (Exploration and Graphics for RivEr Trends) has been developed for carrying out each of these steps in an integrated manner. The package has been designed to easily accept data from three sources: U.S. Geological Survey hydrologic data, U.S. Environmental Protection Agency (EPA) STORET data, and user-supplied flat files. A related R package called dataRetrieval is required in order for EGRET to accomplish these data retrievals for use by the EGRET package, but dataRetrieval also provides the capability to download a broader suite of data types and organize those data in ways that facilitate many kinds of hydrologic applications. The dataRetrieval package is documented in appendix 1 of this report. The EGRET package has components oriented towards the description of long-term changes in streamflow statistics (high flow, average flow, and low flow) as well as changes in water quality. For the water-quality analysis, it uses Weighted Regressions on Time, Discharge and Season (WRTDS) to describe long-term trends in both concentration and flux. EGRET also creates a wide range of graphical presentations of the water-quality data and of the WRTDS results. The body of this report serves as a user guide to the EGRET R package, providing detailed guidance on installation and use of the software, documentation of the analysis methods used, as well as guidance on some of the kinds of questions and approaches that the software can facilitate.

Introduction

The analysis of data about rivers, their flow and their quality, depends on having a set of tools that are appropriate to the nature of the data and to the questions that one wishes to answer. An important component of the analysis tool is having a straightforward method of obtaining the data, checking it, and structuring it in a way that facilitates the analysis. This report is a guide to a tightly coupled system of software for doing both the data retrieval and the data analysis. Two R software packages make up this system (R Core Team, 2013). R is a free software environment for statistical computing and graphics that compiles and runs on a wide variety of UNIX platforms, Windows®, and MacOS.

The first of these packages is known as dataRetrieval. It is designed to obtain water-quality sample data, daily, instantaneous, and metadata directly from the U.S. Geological Survey National Water Information System (USGS NWIS) data services (Hirsch and Fisher, 2014) as well as water-quality data from the Water Quality Portal (Scott and others, 2008). It also allows for user-supplied text files as inputs for each of these data types. The program is designed to load the data directly into R and organize them into file structures suited to the analysis.

The second of these packages is known as EGRET, which stands for Exploration and Graphics for RivEr Trends. It contains its own data-retrieval functions, based on functions in the dataRetrieval package, but which are designed to meet the very specific requirements of the EGRET package. The underlying objective of EGRET is to enable the hydrologist to explore river data for variations in discharge, concentrations of an analyte (such as a major ion, a nutrient, or suspended sediment), and fluxes of an analyte and describe, quantify, and visualize their behavior. It can describe long-term averages, the patterns of variability, as well as temporal trends in these variables. The focus of EGRET can best be described under the general heading of exploratory data analysis (Tukey, 1977) as opposed to statistical inference or hypothesis testing. Within that overall framework, EGRET carries out three types of tasks.

1. The first is referred to as the flow history component of EGRET. It provides a variety of tabular and graphical outputs focused on discharge statistics such as the annual mean, annual 7-day low flow, and annual 1-day maximum, as well as seasonal or monthly versions of these statistics. All of these outputs are based on time-series smoothing methods. It was designed for studies of long-term discharge change that may be focused on questions such as how discharge may be changing because of changes in climate, land use, water use, or water management. It works best for complete daily discharge data sets of 50 years or longer.

2. The second is the graphical display of water-quality sample data as they vary in relation to time, discharge, or season. All of the EGRET functionality for water quality assumes that there is a complete record of daily discharge data for the site where the water-quality data were collected, or from a location sufficiently close to the site so that it provides a good representation of discharge conditions at the site. This discharge record must cover the entire period when the water-quality samples were collected. There must be a date and analyte concentration for each sample in the record. Time of day information is not used in EGRET. The analyte concentrations may be censored values, and these include left-censored data (for example "less than 0.1 mg/L" [milligrams per liter]) and interval-censored data (for example "less than 1.1 mg/L but greater than 1.0 mg/L"). The graphics produced can be very useful for characterizing how the analyte concentration relates to discharge, how it varies by season, and how it may be changing over a period of years. These simple empirical descriptions are often useful to help support results produced by rather complex methods of analysis such as those conducted in the third component of EGRET.

3. This third component of EGRET involves applying the Weighted Regressions on Time, Discharge, and Season (WRTDS) smoothing method (Hirsch and others, 2010) to interpret the behavior of the water-quality analyte of interest on the basis of four components: the relation to discharge, seasonality, long-term trend, and a random component. This analysis produces tabular and graphical representations of the concentration and the flux of the analyte as it relates to these driving factors. It presents annual and seasonal summaries of the behavior of concentration and flux over time and flow-normalized estimates of concentration and flux that are designed to remove the influence of year-to-year variations in discharge and thus provide more insight on underlying changes in the behavior of the watershed. This component also provides a wide range of specialized graphics aimed at identifying potential problems of bias in flux estimates (Hirsch, 2014) and describing the nature of the changes that have taken place. For example, it can identify changes that are specific to base flow or high-flow conditions and (or) particular seasons of the year.

Organization of This Report

This report begins with a section that describes the options for entering each of the three primary data types used by EGRET: discharge, water quality, and metadata. These three data types each have a specific type of data frame that organizes all of the data entered and contains outputs computed from those data. The names of these three data frames are: `Daily` (daily mean discharge data), `Sample` (water-quality data), and `INFO` (metadata). A data frame in R is a two-dimensional matrix (a table), in which each column is a variable (with its own name and data type), and each row is a specific observation. A general understanding of the three primary data frames is very valuable to the user. Knowing what is stored in these data frames and what the variables are named makes it possible for the user to employ other statistical or graphic functionality that exists in R. For each of the three data frames, this report provides an explanation of how to use the EGRET package to enter the relevant discharge data, water-quality data, and metadata. Users who want to work with a wider range of data sets (for example, multiple chemical constituents, sub-daily time steps, time series from water quality sensors) can find relevant functions for these data types in dataRetrieval (documented in appendix 1). The section on data entry is followed by two brief explanatory subsections. The first explains how EGRET handles the definition of seasons and years (called the "Period of Analysis"), and the second explains how EGRET handles units.

The section about data entry is followed by sections covering each of the three components of EGRET discussed above: flow history, summarizing water-quality data (without using WRTDS), and WRTDS analysis of water-quality data. Each of these sections describes the concepts and mathematics involved, identifies the specific EGRET functions that are available in the package, and provides illustrations of the types of outputs they produce. Near the end of the report are shorter sections about editing data sets, working with multiple data frames, and batch processing. In addition there are 4 appendices: 1) An introduction to dataRetrieval (which includes descriptions of several data retrieval functions not related to EGRET as well as those that are used for EGRET), 2) An introduction to EGRET, 3) Example scripts for batch workflows, and 4) A simplified sample workflow.

Getting Help with Functions

It is assumed that the reader of this report has a rudimentary knowledge of the R language and environment. General information on R can be obtained from the R-project at *http://www.r-project.org/* (particularly the Manuals page). Many texts also teach about using R.

Detailed directions on using individual functions, with the full range of user options, are presented in appendixes 1 and 2 for the dataRetrieval and EGRET packages, respectively. These appendixes are also known as "vignettes," and they follow a standard format used for R packages. In addition, very detailed descriptions of all of the functions contained in both packages are available to the user within the R software environment once the dataRetrieval and EGRET packages have been loaded onto the user's computer. For example, information about the function `readNWISSample` in the dataRetrieval package can be obtained with the command `?readNWISSample`.

Users will note from the appendixes that most of the functions in dataRetrieval and EGRET have a very large number of arguments, most of which have default values. This is particularly true for all of the graphics functions, which are designed to offer a great deal of flexibility in terms of size of characters and setup of axes and their labels. However, they were all designed in such a way that the default settings create a highly presentable graphic for reports and presentations. The body of this user guide does not delve into most of these arguments. Users who want to make adjustments are directed to the help pages and the EGRET vignette (appendix 2) for instructions for using the arguments. In most cases, a function that produces a graph or a table can be used with only a single argument.

Examples of commands given in this report generally specify a minimal set of arguments and ignore many of the arguments used for detailed settings of graphics. For most of the functions, the first argument is the name of a list which specifies the four objects that contain all of the data and statistical model outputs. In general, this list is called `eList`. In many cases the additional arguments are for setting details of the graphical output or model estimation parameters. Generally the functions can be called with all of those arguments set to their default values; thus, the arguments do not need to be specified in the actual command line. Advanced users may want to create alternative versions of these and other data frames to make various types of comparisons. This kind of advanced use is described in the section of the report entitled "Working with Multiple Versions of Data Frames."

The information in this report and the appendixes is written to describe the interactive use of the software. However, as is generally the case in R, batch processing is easily done by creating a file of specific commands to be executed. Once the proper workflow for a particular project is established, this file of instructions can be input to the R environment on the user's computer and the computations can proceed without the need for repeated steps by the user. Suggestions for batch processing are provided in a section titled "Batch Processing in EGRET" near the end of the user guide. Finally, there is a short section showing a "Sample Workflow" for flowHistory and WRTDS analysis. These can serve as "ready reference" materials to remind users of the basic flow of the processes and the names and usage of the most important functions. Another important feature of R is that all of the code in the basic R package, as well as the two specific packages described here, is freely available to the user. To see the code for the function `readNWISSample`, for example, the user simply can enter `readNWISSample` (without parentheses) at the R-command line and the complete code will appear on the user's console, from which it can be freely copied. This means that users who want to develop their own R applications that have some similarity to those in the dataRetrieval and EGRET packages can examine the code and then copy and modify it to suit their own needs. There are no copyright restrictions on any of the basic R codes or any aspects of these two packages.

Package Installation

To install the dataRetrieval and EGRET packages or updates to the packages, the command is:

```
install.packages(c("dataRetrieval","EGRET"))
```

Users of EGRET will need to make sure that their version of dataRetrieval is up to date, thus, both should be installed at the same time with this single command. If installing an updated version, it is a good idea to restart R after installing the packages. Once the package has been installed, you will need to open the library each time you restart R. This is done with the simple command:

```
library(EGRET)
```

Data Entry

In the examples to follow, we consider a streamgage (Choptank River near Greensboro, Maryland) with USGS site ID 01491000, and we look for daily mean discharge and sample values of dissolved nitrate plus nitrite, which has parameter code 00631. Because data discovery is not the focus of this software or report, it is assumed that the user already knows the unique identifier of the streamgage and the desired parameter code. A number of data discovery tools can facilitate finding sites within a region that may meet the user's data requirements. For example, the NWISWeb system (*http://waterdata.usgs.gov/nwis*) allows a user to discover sites with a specified minimum amount of data (such as a minimum number of daily mean streamflow values or a minimum number of water-quality samples). The dataRetrieval package provides a capability to use several types of data sources. The body of this report focuses only on those functions needed to retrieve data for the EGRET package.

Daily Mean Discharge Data for Use in EGRET

The EGRET software requires that there be a continuous record of daily mean discharge data for the study period. In those cases where only flow history is being analyzed, the user may want to obtain the entire record, or if the study period has a standard starting and ending date, then those dates can be used. In the case of a WRTDS water-quality study, the period for which the WRTDS model will be estimated is defined entirely by the user's selection of the starting and ending dates of the daily discharge data, which must completely span the period of record of the water-quality data that will be used in the analysis. In other words, the discharge record being used must start on or before the first sample day, and end on or after the last sample day. However, the daily discharge record should not extend more than a few months beyond the range of the water-quality record. Using a discharge record that extends far beyond the period of water-quality record will cause WRTDS to perform extrapolations. WRTDS is a smoothing method and, as such cannot be expected to produce meaningful extrapolations in time; estimates for dates a few years beyond the water-quality record can be unrealistic. Thus, a good practice would be to select a starting date that is the beginning of the first water year in which there are water-quality data and an ending date that is the last day of the last water year in which there are data. Starting and ending dates need not be at the start or end of the water year, but the user's choice of these dates, and the choice of the whether annual results are to be presented on a water year, calendar year, or seasonal basis, will influence whether or not the first and (or) last years of the analysis will be reported.

The daily mean discharge data are stored in EGRET in a data frame called `Daily`. It has a very specific structure and naming convention for all of its columns, although users can add columns as needed. Table 1 describes the contents of the `Daily` data frame. The number of rows in the data frame is equal to the number of days in the daily mean discharge record. Initially, the number of columns is 12. However, once the WRTDS computations have been run by using the `modelEstimation` function, the data frame is automatically augmented with an additional six columns computed by that function (as described in the section "WRTDS Analysis of Water-Quality Data").

Table 1. A list of the column names for the `Daily` data frame.

[Column names shown in black are created when the information is retrieved and stored when the data frame is created. Column names shown in red are created and stored automatically when the Weighted Regressions on Time, Discharge, and Season (WRTDS) computations are made by the EGRET package by using the modelEstimation function]

Column name	Definition	Data type	Units
`Date`	The date	Date	yyyy-mm-dd
`Q`	Daily mean discharge on that date	Numeric	m^3/s
`Julian`	The date expressed as days starting with Jan. 1, 1850	Numeric	Days (integer)
`Month`	Month of the year, from 1 to 12	Numeric	Months (integer)
`Day`	Day of the year, from 1 to 366	Numeric	Days (integer)
`DecYear`	Year expressed as a decimal	Numeric	Years
`MonthSeq`	Month sequence: an index starting with 1 at Jan. 1850	Numeric	Months (integer)
`Qualifier`	USGS qualification code, A = Approved, P=Provisional	Character	-
`i`	Index value of days, from the first day in the data frame	Numeric	Days (integer)
`LogQ`	ln(Q)	Numeric	Dimensionless
`Q7`	Mean discharge for 7 days, up to day i	Numeric	m^3/s
`Q30`	Mean discharge for 30 days, up to day i	Numeric	m^3/s

Table 1. A list of the column names for the Daily data frame.—Continued

[Column names shown in black are created when the information is retrieved and stored when the data frame is created. Column names shown in red are created and stored automatically when the Weighted Regressions on Time, Discharge, and Season (WRTDS) computations are made by the EGRET package by using the modelEstimation function]

Column name	Definition	Data type	Units
yHat	The WRTDS estimate of the log of concentration	Numeric	Dimensionless
SE	The WRTDS estimate of the standard error of yHat	Numeric	Dimensionless
ConcDay	The WRTDS estimate of concentration	Numeric	mg/L
FluxDay	The WRTDS estimate of flux	Numeric	kg/day
FNConc	Flow-normalized estimate of concentration	Numeric	mg/L
FNFlux	Flow-normalized estimate of flux	Numeric	kg/day

The variable Q7 is the mean discharge for the 7 days up to and including the specific day for that row (for example, for September 21, 2010, Q7 would be the average of the discharge values for September 15–21, 2010). Q30 is the mean for the 30 days up to and including the specific day for that row. Note that at the start of the record, there are 6 missing values for Q7 (designated as NA) and 29 missing values for Q30. There are two ways to obtain the discharge data: either from USGS data services or from a user-supplied data file.

Discharge Data from the U.S. Geological Survey Data Service

For those cases where the discharge data are from the USGS, obtaining the data from a Web service retrieval is the preferred approach. The source for these data is the USGS data service at *http://waterservices.usgs.gov/*.

The function used to obtain the daily discharge data is readNWISDaily. In the case of the Choptank River example, the set of commands would be as follows:

```
siteNumber <- "01491000"

QParameterCd <- "00060"

StartDate <- "1979-10-01"

EndDate <- "2012-09-30"

Daily <- readNWISDaily(siteNumber, QParameterCd, StartDate, EndDate)
```

Alternatively, one could proceed without defining the four arguments to the function and enter them directly into the function call, which would then look like this:

```
Daily <-readNWISDaily("01491000","00060","1979-10-01","2010-09-30")
```

This command instructs the user's computer to retrieve the daily mean discharge data from a specified USGS URL and make the necessary computations to create the Daily data frame for the specified streamgage and period of record. The command also returns some information about the length of the record, gaps that may exist, and the presence of zero or negative discharge days. The second argument value in the readNWISDaily function for an EGRET application should always be "00060" (which is discharge in cubic feet per second). readNWISDaily converts the data in cubic feet per second to cubic meters per second by default. The dataRetrieval package contains a very similar function called readNWISdv, but it is designed so that it can be used to retrieve other parameters that are stored in a daily values format for analyses that are outside the scope of the EGRET package. In such cases if one is using a different parameter code than "00060", be sure also to include the argument convert=FALSE in the command, otherwise a conversion factor is applied. A list of some common parameter codes for daily data available from the USGS are given in appendix 1, table 1. Users interested in obtaining other types of daily data should refer to appendix 1.

If the user is uncertain about what the starting and ending dates should be, or wants to retrieve the entire period of record, the command could be:

```
Daily <- readNWISDaily("01491000","00060",startDate="",endDate="")
```

This results in a record that contains all available daily mean discharge data for the requested streamgage. It is not uncommon for the final few weeks or months of the record to contain missing values. The program returns information to the console telling the number of days between the first and last days of the record and the number of daily values . If the number of daily values is smaller than this time span, then the program also states the starting and ending date of the gaps. This information can help the user determine how to shorten the total period requested to obtain a record without gaps. If there are gaps, the user must modify the `startDate` and (or) `endDate` to avoid gaps in the record. In a water-quality data analysis project, it may be necessary for the user to iterate between the discharge data retrieval and the water-quality sample retrieval to obtain an appropriate period of record for both. This iteration requires rerunning the `readNWISDaily` function. Each time the function is called, it simply overwrites the previous version of the `Daily` data frame with the new one. The user does not need to delete the earlier version of `Daily`.

Data in the record may be denoted as "P" (Provisional) or "A" (Approved) in the `Daily$Qualifier` column. The more recent portion of the record may contain provisional data, whereas the older data have passed the USGS approval process. The following series of commands enables the user to identify the span of dates for which the reported data are Provisional.

```
DQ <- subset(Daily,Qualifier=="P")
```

This command simply creates a new data frame from the `Daily` data frame, selecting only those days for which the Qualifier is "P."

```
summary(DQ$Date)
```

This command returns a summary of the dates in `DQ` showing the first and last provisional date. If the user does not want to use provisional discharge data, then entering the following command removes the rows with provisional data:

```
Daily <- subset(Daily,Qualifier!="P")
```

The user may want to clean up their workspace at this point by removing the data frame `DQ`. The command is:

```
rm(DQ)
```

The `readNWISDaily` function also reports the number of days of zero or negative discharge. Negative discharge values can arise at sites with backwater conditions. Because many of the computations in EGRET use the natural log of discharge, there cannot be any zero or negative discharge days in the record. The presence of even one zero or negative discharge value causes the EGRET calculations to fail. If there are any zero or negative values, they are all be set to zero and then a very small constant is added to all of the discharge data. This constant is set to 0.1 percent of the mean discharge in the record. If there is a very small number of zero or negative flow days (for example less than 0.2 percent of the days), then there should be no serious problems in using EGRET. But if there is large numbers of zero or negative flow days, then the statistical methods will probably be compromised by having a large number of days all having values tied at this arbitrary minimum value of discharge. Final numerical results from EGRET should have this small flow increment subtracted out of discharge and flux results, but typically, these adjustments would be so small as to have no consequence if users are reporting results to an appropriate number of significant figures. For sites that commonly have zero or negative discharge values the methods used in EGRET are generally inappropriate although the dataRetrieval functions can provide a useful means of obtaining and organizing data for some types of analysis.

Once the data are retrieved, the data set can be explored with the command:

```
summary(Daily)
```

It reports the time span of the data set and the range of the data values. Note that the discharge values in the summary are automatically converted to cubic meter per second (m^3/s), the discharge unit used for all calculations in EGRET. For all of the specific graphical and tabular output functions of EGRET, however, the user has a choice of four different units in which to report discharge (see subsequent section "Selecting Units of Measurement for Graphs and Tables in EGRET").

The total number of daily mean discharge values can be obtained by giving the command:

```
length(Daily$Q)
```

In addition, a very simple plot of the time series can be obtained with the command:

```
plot(Daily$DecYear,Daily$Q,log="y",type="l")
```

Note, in type=l that the l is a lower case letter L, which stands for "line." When type is not listed, the plot function's default symbol is a point rather than a line. Looking at the summary and at such a time series plot is a good opportunity to spot data problems that may need to be resolved.

Discharge Data from a Text File

If the discharge data are not available from USGS data services, but they are available in the form of a spreadsheet, then they can be entered as comma-separated values from a file (usually denoted with a ".csv" extension). The option of using other file types is discussed in more detail in appendix 1. The file must consist of two columns: the first is the date expressed as mm/dd/yyyy or yyyy-mm-dd, and the second is daily discharge. A common error is the use of a two-digit year (for example, 04/25/12). Spreadsheet programs tend to revert to these two-digit years, even though they are ambiguous. The first row in the file should contain headings such as "date" and "Qdaily," although the choice of names is of no consequence.

The user must define two variables to identify the full name of the data file: the first of these is `filePath`, and the second is `fileName`. The variable `filePath` is a character string that defines the path to the file on the user's computer. This can either be a full path name, or path relative to the R working directory, and should end with "/". Users of Windows® operating systems should be aware that R requires that paths use the forward slash ("/"), whereas Windows® tends to use the backwards slash ("\") in many other applications. Using the backwards slash results in an error. Regardless of the operating system used, the forward slash is required in R. The variable `fileName` is a string that defines the file name (including the extension).

As an example, for the MacOS:

```
filePath <- "/Users/rhirsch/desktop/"
fileName <- "ChoptankFlow.csv"
```

Once those two variables are defined, the function can be called. In its simplest form it would be:

```
Daily <- readUserDaily(filePath, fileName)
```

The default case is that the discharge values in the file are in units of cubic foot per second (ft^3/s). The alternative is that they are expressed in m^3/s, and if that were the case then the command would be:

```
Daily <- readUserDaily(filePath, fileName, qUnit = 2)
```

If the spreadsheet available provides discharge in some other units, for example, liter per second (L/s) or gallon per hour (gal/hr), then the user must make a conversion in their spreadsheet program to convert them to either m^3/s or ft^3/s. The same rules regarding the starting and ending date of the record apply here as were described above, but they are set simply by the user editing the data file to have the appropriate starting and ending dates. The function `readUserDaily` checks for zero and negative discharge values and makes the necessary changes to the data set as are described above for the function `readNWISDaily`.

Water-Quality Data for Use in EGRET

Water-quality data are stored in EGRET in a data frame called `Sample`. It contains information about only one analyte, although it may be that this analyte is computed as the sum of two or more analytes (see discussion below). The `Sample` data frame has a specific structure and naming convention for all of its columns, although users can always add columns, and table 2 describes its contents. The number of rows in the data frame is equal to the number of sample values in the data set. Initially, the number of columns is 14. However, once the WRTDS computations have been run by using the `modelEstimation` function, the data frame is automatically augmented with an additional three columns computed by that function (as described below in the section "WRTDS analysis of water quality data").

Table 2. A list of column names for the `Sample` data frame. Those shown in black are computed and stored when the data frame is created. Those in red are created and stored automatically when the Weighted Regressions on Time, Discharge, and Season (WRTDS) computations are made by the EGRET package by using the `modelEstimation` function.

Name	Definition	Data type	Units
`Date`	The date of the sample	Date	yyyy-mm-dd
`ConcLow`	Lower bound for an observed concentration	Numeric	mg/L
`ConcHigh`	Upper bound for an observed concentration	Numeric	mg/L
`Uncen`	=1 if sample is uncensored, =0 if censored	Numeric	Integer
`ConcAve`	Average of `ConcLow` and `ConcHigh`	Numeric	mg/L
`Julian`	The date expressed as days starting with Jan. 1, 1850	Numeric	Days (integer)
`Month`	Month of the year, from 1 to 12	Numeric	Months (integer)
`Day`	Day of the year	Numeric	Days (integer)
`DecYear`	Year expressed as a decimal	Numeric	Years
`MonthSeq`	Month sequence: an index starting with 1 at Jan. 1850	Numeric	Months (integer)
`SinDY`	sin(2*π*DecYear)	Numeric	Dimensionless
`CosDY`	cos(2*π*DecYear)	Numeric	Dimensionless
`Q`	Daily mean discharge on the day of observation	Numeric	m^3/s
`LogQ`	Natural logarithm of Q	Numeric	Dimensionless
`yHat`	Cross-validation estimate of the log of concentration	Numeric	Dimensionless
`SE`	Cross-validation estimate of the standard error of `yHat`	Numeric	Dimensionless
`ConcHat`	Cross-validation estimate of concentration	Numeric	mg/L

The `Sample` data frame is designed to accommodate censored data. Typically, when the concentration is very close to zero, the laboratory reports the concentration as "less than" a reporting limit. In the USGS NWIS database, this is communicated through a remark code of "<" followed by a value that is typically set to the reporting limit. The dataRetrieval package structures the concentration data in the `Sample` data frame in a manner that makes them suitable for use in the survival regression, which is the statistical method used by WRTDS. The concentration information is stored using two variables, one called `ConcLow` and the other called `ConcHigh`. When the data are uncensored, these two variables are equal to each other (set to the reported concentration value). If a value is censored, for example with a reporting limit of 0.5 mg/L, then `ConcLow` is set to `NA` (which stands for Not Available) and `ConcHigh` is set at the reporting limit, in this case 0.5. The variable called `Uncen` is always 1 for an uncensored value and 0 for a censored one. The mean value of `Uncen` for any given data set is equal to the frequency of uncensored values in the data set; for example, if the mean of `Uncen` is 0.95, then 95 percent of the values are uncensored and thus 5 percent of the values are censored. The variable `ConcAve` is provided for convenience and is used in some of the graphics functions but is not used in any computation. For censored values where `ConcLow` is equal to `NA`, `ConcAve` it is set equal to `0.5*ConcHigh`. This type of censoring is referred to in the statistical literature as "left censoring."

The `Sample` data frame structure is also designed to handle the more complex case of interval censoring. The following is an example of one of the ways that interval censoring can arise. In the Chesapeake Bay River Input Monitoring Program (Moyer and others, 2012, p. 4,6,9–11), there is a set of rules for computing total nitrogen as the sum of multiple individual nitrogen analytes. The rule that applies to the data collected at the Potomac River at Chain Bridge, Washington, D.C. for the year 1999 is that total nitrogen is computed as the sum of two analytes that were measured at that time: one is nitrate plus nitrite, filtered (USGS parameter code 00631) and the other is ammonia plus organic nitrogen, unfiltered (USGS parameter code 00625). Consider the sample from June 7, 1999. The nitrate plus nitrite was reported as 0.596 mg/L, and the ammonia plus organic nitrogen was reported as <0.1 mg/L. The sum of these two values lies in the range 0.569 mg/L to 0.696 mg/L. By using the convention adopted for the `Sample` data frame, the `ConcLow` value would be 0.569, the `ConcHigh` value would be 0.696, and the variable `Uncen` would be equal to 0. Because this situation (where two or more analytes are summed to form the variable of interest) is somewhat uncommon, the details of data entry for these cases are presented in appendix 2, section 3.2.4, rather than here in the body of the report.

There are two other instances where the interval censoring approach may prove useful. One is dealing with changes in rounding of reported concentrations. It is not uncommon to see a set of concentration data in which those in the later part of the record are reported with more significant figures than those in the earlier part of the record. If the user is concerned that the

change in rounding practices may influence the results of the subsequent analysis, it may be useful to code the more rounded values to be represented as interval censored. For example, consider a case where the data are reported as 0.01, <0.01, 0.03, and 0.02, but after the fourth value the laboratory changes its reporting conventions so that more significant figures are reported and the next three values in the same data set are reported as 0.0134, 0.0545, 0.0252. By using the reporting conventions used in EGRET, the user might instead report these seven observations as follows so that each is represented by a `ConcLow`, `ConcHigh` pair of values: (0.005, 0.015), (NA, 0.01), (0.025, 0.035), (0.015,0.025), (0.0134,0.0134), (0.0545,0.0545), and (0.0252,0.0252). The process of creating these intervals is described below in the section "Editing Data Sets."

Yet another application of interval censoring could be used to modify concentration values for which there is an unusually large amount of uncertainty; for example, a measurement made in flood conditions where the sample collection method could have resulted in a biased sample. If the user is prepared to indicate an upper and lower bound on what might be the true value for the sample, then they can be substituted for `ConcLow` and `ConcHigh`. Using these interval estimates can be a useful sensitivity check for learning if a few highly uncertain, but highly important, samples could have a large influence on the final analysis of long-term fluxes or trends. These types of changes are also described below in the section "Editing Data Sets."

Water-Quality Data from U.S. Geological Survey Web Services

To obtain the water-quality data from the USGS Data Services, a station number, parameter code, and a starting and ending date are needed. The function used to obtain the data is `readNWISSample`. In the case of the Choptank River, for the parameter nitrate plus nitrite, filtered, reported as N, (parameter code 00631), and assuming that we want sample values from water years 1980 through 2012, the set of commands could be as follows:

```
siteNumber<- "01491000"

ParameterCd <- "00631"

StartDate <- "1979-10-01"

EndDate <- "2012-09-30"

Sample <- readNWISSample(siteNumber, ParameterCd, StartDate, EndDate)
```

or, alternatively, one could proceed without defining the four arguments to the function and enter them directly into the function call, which would then look like this:

```
Sample <- readNWISSample("01491000","00631","1979-10-01","2010-09-30")
```

Another approach, when the user is unsure of the time period for which sample data are available, would be to give the command in this manner:

```
Sample <- readNWISSample(siteNumber, ParameterCd, startDate="", endDate="")
```

This returns all data for that site and parameter. To determine what the first and last sample dates are, the user can give the command:

```
summary(Sample)
```

The first column will show the first and last dates. Note that at this stage of the process, there will be no columns in the `Sample` data frame for `Q` or `LogQ`. These columns are created later by using the function called `mergeReport`, which obtains the discharge data from the `Daily` data frame.

There are times when the data set has just a few values in the early years and then denser sampling commences later. The user may wish to eliminate the early part of the record from the analysis because the data are so sparse or a gap is too long. An easy way to identify this type of situation is with a simple plot.

```
plot(Sample$DecYear,Sample$ConcHigh)
```

This plot will show if there are large gaps in the sampling record, if so, then the user might choose to adjust the request by choosing a more restrictive set of values for `startDate` and `endDate`. Depending on the dates chosen, it may be necessary to return to the `readNWISDaily` function to obtain a discharge data set that fully covers the sample data time period but that does not extend far beyond the first and last sample dates. Large data gaps (such as two or more years of no data) should be noted. The WRTDS analysis can be run on data sets with gaps of several years in the water-quality record. The results during the gap period will be highly unreliable, but the results on either side of the gap period will be quite reliable. Once the WRTDS analysis is completed (by using `modelEstimation`), the results can be modified to remove the results from the data gap period by using the function `blankTime,` which is described in the section on WRTDS analysis.

Water-Quality Data from the Water Quality Portal

There are additional water quality data sets available from the Water Quality Data Portal (*http://www.waterqualitydata.us/*). These data sets are housed in: the STORET database (data from the Environmental Protection Agency (EPA)), NWIS database (data from USGS), or STEWARDS database (data from USDA). Additional databases are expected to be included in the future. Because only the USGS uses the 5-digit parameter codes, a "characteristic name" must be supplied. The `readWQPSample` function can take either a USGS parameter code, or a more general characteristic name in the `parameterCd` input argument. The Water Quality Data Portal includes data discovery tools and information on characteristic names. The following example retrieves specific conductance from a Wisconsin Department of Natural Resources monitoring site.

```
    Sample <- readWQPSample("WIDNR_WQX-10032762","Specific conductance", "2011-05-01",
"2011-09-30")
```

Guidance for finding characteristic names can be found at:*http://www.waterqualitydata.us/webservices_documentation.jsp*.

Water-Quality Data from a User-supplied File

If the water-quality data are not available from one of these data services, they can be entered from a user-supplied file. Just as in the case of the discharge data from a user-supplied file, the user will have to supply a specific `filePath` and `fileName`. For example, on a MacOS these could be:

```
    filePath<-"/Users/rhirsch/Desktop/waterqualitydata/"
    fileName<-"ChopNO3"
```

The default structure of the file is a csv file, although other structures such as tab delimited or space delimited are possible. The default structure also expects column headings for each column. Alternatives to this default structure are described in appendix 2, section 3.2 and in the help for `readUserSample`. In the default case, each column needs a header in the first row. Headers can be named something like "date," "remark," and "value", but alternative names will work. The first column must contain the sample date, which should be in either mm/dd/yyyy or yyyy-mm-dd. Two-digit years are not acceptable because they are ambiguous. The second column is for the remark, and the column is required even if there are no remarks. This column should show "<" for all less-than values and be blank otherwise. The third column is the concentration. Where there is a "<" in the remark column, the concentration column should contain the reporting limit. It is assumed that all concentrations are in mg/L. There should never be a zero value in the concentration column; the function will look for that, and if it exists, a warning is displayed and the value is discarded. If this situation arises, the original data set should be reviewed and a proper interpretation of the zero values should be determined, if possible. They should be recoded in the original data set, preferably to have a "<" in the remark column and the zero replaced with a reporting limit in the value column. In cases where the user cannot determine what the correct reporting limit should be, some reasonable convention may be used, such as setting these values to some number less than the lowest reported value. This type of approximation is likely to be better than leaving the value at zero and or deleting the observation, because these alternatives can result in a serious bias. In the more complex case, where the variable of interest is the sum of multiple analytes, the input file can contain more than three columns. For each additional analyte there is a remark column followed by a value column. The function sums analytes across all of the columns to come up with the concentration value that is used. The presence of these additional pairs of remark and concentration values causes the program to compute concentrations as the sum of the analytes included. For details, refer to section 3.2.4 of appendix 2.

Entering and Storing Metadata

Metadata about the site and the analyte being evaluated are stored in a data frame named INFO. Depending on how metadata are acquired, there can be a large number of data elements in INFO. These are listed in appendix 1. Only a few items of metadata are required for the EGRET functions to run (table 3).

Table 3. Variables required in the INFO data frame for use in EGRET applications.

Variable name	Definition	Purpose
shortName	Name of data collection site	Name of site used in all graphs and tables.
paramShortName	Name of the parameter (or constituent)	Name of parameter as used in all graphs and tables.
staAbbrev	Abbreviated name of site	Abbreviation used to name files storing workspaces for the site.
constitAbbrev	Abbreviated name of parameter (or constituent)	Abbreviation used to name files storing workspaces for the parameter (or constituent).
drainSqKm	Drainage area at the monitoring site in km^2.	Used to compute runoff (for example, in mm/d) or yields (for example, in $kg/y/km^2$).

For all situations, except those where the water-quality data come from the Water Quality Portal, the metadata are entered into the system by using the function readNWISInfo. If all the data come from a USGS data service call, the command for USGS site number 01491000 and parameter number 00631 would look like this:

```
INFO <- readNWISInfo(siteNumber = "01491000", parameterCd ="00631")
```

Or equivalently

```
INFO <- readNWISInfo("01491000", "00631")
```

If the application deals only with discharge data and not water-quality data, the command would be:

```
INFO <- readNWISInfo(siteNumber = "01491000", parameterCd = "00060")
```

If the water quality data came from the Water Quality Portal, then siteNumber would be either in the form "USGS-10491000" (if it is a USGS site) or in the form "WIDNR_WQX-10032762" (if it is from a non-USGS source, the initial characters before the "_" are an agency identifier, in this case the Wisconsin Department of Natural Resources). The parameterCd would be either a USGS parameter code or a "characteristic name." So for an example of specific conductance data, the command might be:

```
INFO <- readWQPInfo("WIDNR_WQX-10032762","Specific conductance")
```

In a case where the discharge data came from the USGS data service but the water-quality data came from a user-supplied file, the command would be in the form:

```
INFO <- readNWISInfo(siteNumber="01491000",parameterCd = "")
```

In this case, the program will prompt the user for parameter information.

If all of the data came from user supplied files, then the command would be:

```
INFO <- readNWISInfo("", "")
```

In this case, the program will prompt the user to supply all of the metadata. When the user supplies metadata, the only elements that they must supply are those listed in table 3 and the last one could be entered as NA.

Users can always supply additional information to be stored in the INFO data frame. To do this, the user defines a new variable to be a part of the INFO data frame and assigns it a value. For example, if we wanted to store the fact that there was a switch from plastic to glass sample bottles on 2005-10-01, then after INFO has been created we can give the command:

```
INFO$bottleNote <- "switched from plastic to glass bottles 2005-10-01"
```

The readNWISInfo function populates the INFO data frame from the metadata that it acquires from USGS data services, where that is possible, by using the station number and parameter code. In general, this is an interactive process, and readNWISInfo provides to the user a series of prompts requesting user input. The specific prompts depend on the information provided in the call to the function. The following are some of the items that are requested and the reasons for them.

- Station name: When a station number (siteNumber) is supplied, the prompt provides the user with the exact name of the station in the USGS database. First, the user should check to see if this is actually the intended site. Assuming it is, the user has the option to enter the station name in a different form, perhaps chosen to provide the most readable and informative rendition of the name as it will appear on all figures and tables. This option is available because, in some cases, the station name as provided is not particularly suitable for use in figure or table titles. For example, some stations have names that are in all capital letters, which can be difficult to read. The user can instead type in a version that uses upper and lower case letters. Sometimes the site name is exceedingly long, and the user may want to shorten it to make it more suitable for use in graphs and tables. If the user is satisfied with the name as provided, all that is needed is a carriage return and the name is used. The original official USGS station name is retained in the INFO data frame so that if there is any doubt about the official name, it remains available.

- Parameter names: Similarly, parameter names can be very long and complex, and thus, not suitable for use in titles of figures or tables. If the parameter code is in the call to readNWISInfo, the full parameter name is provided, and then the user is prompted to provide a shorter alternative more suitable for use in figures and tables. If the original wording is acceptable, all that is needed is a carriage return. Again, the official name of the parameter is retained in the INFO data frame.

- Station and parameter abbreviations: The readNWISInfo function prompts the user for abbreviations. These can be very helpful to the user for managing the various files involved in a project with many sites and parameters. The user should develop their own lexicon of simple abbreviations for their sites; for example, the Choptank site could be "Chop" and the nitrate parameter could be "NO3". Files that the user may create and update in the course of the project are automatically named through the use of these abbreviations. (See the section "Saving the Workspace for Future Use"). So, for example the workspace containing the Choptank River nitrate example would be called "Chop.NO3.Rdata." This can be helpful when structuring large batch jobs as well as for returning to a previous data analysis.

- Drainage area: If the site is in the USGS database, then a drainage area should be stored. It will typically be in square miles (mi^2). The prompt gives this information and shows the value converted to km^2 (both numbers are stored). But if there is no known drainage area, the user is prompted to provide one and the units in which it is provided (mi^2, km^2, acres, or hectares). If the user has no way of knowing the drainage area, enter the number 0 . As a consequence, attempts to compute runoff in units of millimeters per day (mm/day) will fail (as they should) and generate an error message.

There are many additional metadata elements that are stored in INFO if they are available from the user or from data services. In R, typing the name of an object results in the entire content of the object being sent to the user's console, thus users can review all the metadata that are stored in INFO just by using the command:

```
INFO
```

Removing Duplicate Observations

There are cases where there may be multiple observations from a single day that have identical sample values and the user wishes to remove the duplicates. The call to the function that would carry out this process is:

```
Sample <- removeDuplicates(Sample)
```

This command edits the Sample data frame to eliminate such duplicates.

Moving Discharge Data from the Daily Data Frame to the Sample Data Frame

All graphics and analysis in EGRET that relate to water quality require that every sample value is associated with a daily mean discharge value for the day when the sample was taken. To assure consistency between the `Daily` and `Sample` data frames, these data are imported from the `Daily` data frame to the `Sample` data frame. The function that accomplishes this is `mergeReport`. After running `mergeReport` the `Sample` data frame is augmented by the addition of two columns: `Q` and `LogQ`. This function does two additional things: (1) it produces a compact table of information about the content of the `Sample` and `Daily` data frames that can be helpful in spotting potential data problems and (2) it organizes the three data frames (`INFO`, `Daily`, and `Sample`) into a named list called `eList`, which is an object that the code understands to be an EGRET object. The call to the function is:

```
eList <- mergeReport(INFO,Daily,Sample)
```

Figure 1 is an image of the report provided in the Choptank nitrate example.

```
Discharge Record is 12631 days long, which is 35 years
First day of the discharge record is 1978-10-01 and last day is 2013-04-30
The water quality record has 632 samples
The first sample is from 1979-09-25 and the last sample is from 2013-04-29
Discharge: Minimum, mean and maximum 0.00991 4.13 246
Concentration: Minimum, mean and maximum 0.04 1.1 2.4
Percentage of the sample values that are censored is 0.32 %
```

Figure 1. Example of the output from the `mergeReport` function.

From this point forward in the use of EGRET, the user typically supplies `eList` as an argument to each of the functions and the function then uses information from one or more of these data frames to do its computations. Users can always "unpack" the `eList` to see the content of the individual data frames within it. For example, to see a summary of `Sample` after `mergeReport` has been run, the user would give these two commands:

```
Sample <- eList$Sample
summary(Sample)
```

or, alternatively, this can be done with a single command:

```
summary(eList$Sample)
```

Saving the Workspace for Future Use

After the `Daily`, `Sample`, and `INFO` data frames and the object `eList` have been created, it is recommended that the user save the workspace for future use in the EGRET software by defining the variable `savePath`, which indicates the full pathname in the computer's file structure where the workspace will be stored. The user must supply that pathname. For example, on a MacOS these could be:

```
savePath <-"/Users/rhirsch/Desktop/myWorkspaces/"
```

Note the "/" at the end of the name is required. Then, to execute the command and save the workspace, the command would be:

```
saveResults(savePath,eList)
```

The result of saving a workspace by using the INFO data frame variables staAbbrev and constitAbbrev (table 3) for the example site is a file called: /Users/rhirsch/Desktop/myWorkspaces/Chop.NO3.RData. At any time in the future, this workspace can be restored simply by giving the command:

```
load("/Users/rhirsch/Desktop/myWorkspaces/Chop.NO3.RData")
```

Depending on the operating system or specific version of R being used, this can alternately be accomplished from a drop-down menu or by dragging the icon for the file into the console. Users can experiment to find the simplest ways to do this for their implementation of R. During further analysis of the data set, users should use the saveResults function repeatedly so that the completed analysis can be stored for future use. In some cases, users may wish to save different versions of the same data set. The versions may differ in terms of the length of the data set used, some change in the handling of censored values, or some differences in the settings used in conducting certain analyses. These separate versions of the workspace can be stored by creating new abbreviations and then storing the workspace again by using the saveResults function. The use of alternative data frames and lists is discussed in the section titled "Working with multiple versions of data frames."

Setting the Period of Analysis for Graphs, Tables, and Analyses in EGRET

Many hydrologic studies, whether of streamflow or of water quality, need to provide results that are specific to a particular part of the year rather than the entire year. Also, for different audiences, the representation of a year that they are accustomed to seeing may differ. Some audiences want to see the water year (October through September), while others may want to see the calendar year. Throughout the EGRET software, both for flow history studies as well as water-quality studies, the necessary flexibility to show different seasons or different definitions of a year is provided through the concept of the "period of analysis" or "PA." If the outputs are reported by water year, then the PA is October through September. If the outputs are calendar years, then the PA is January through December. If the output is for the winter season, as defined by the months of December, January and February, then those 3 months become the PA. If the outputs are only for the month of May, then the PA is May. The only constraints on the definition of a PA are these: 1) it must be defined in terms of whole months; 2) it must be a set of contiguous months (like March-April-May); 3) it must have a length that is no less than 1 month and no more than 12 months. The PA is uniquely defined by two arguments: paLong and paStart. paLong is the length of the period of analysis in months, and paStart is the first month of the PA (where January is month 1). Table 4 summarizes paLong and paStart through a series of common examples.

Table 4. Examples of the period of analysis and the paStart and paLong values associated with them.

Period of analysis	paStart	paLong
Calendar year	1	12
Water year	10	12
Winter (December–February)	12	3
Warm season (April–September)	4	6
September	9	1

Virtually all of the EGRET functions that provide graphs or tables of the data or statistical results have the capability to restrict their output to be related to some particular PA. The only exceptions to this are the functions plotConcQSmooth, plotConcTimeSmooth, plotContours, and plotDiffContours. The defaults are always paStart = 10 and paLong = 12 (the water year), and all graphics and tables produced by EGRET that use a PA will indicate the PA that was used to produce them.

The year listed in any tabular output from EGRET is the calendar year at the time the PA ends. This is the same approach used for numbering water years. The water year starting October 1, 1990, is water year 1991. If the PA were paLong = 6, paStart = 9 (a period from September through February), the results for the period September 1, 2000, through February 28, 2001, are listed as the 2001 value.

The annual values shown on any EGRET graphic are always plotted at the mean date for the period (expressed in decimal years). Thus, a calendar year average plots at the mean date or midpoint of the calendar year; for calendar year 1981, the mean value plots at 1981.5. The mean value for a water year plots its mean date; for water year 1981, the mean plots at 1981.25. If

the PA were paLong = 4 and paStart = 3 (a period consisting of the months of March through June), then the mean value for 1981 plots at 1981.33 (which is May 1, 1981). The tick mark on the graph for a given year is on January 1 of that year. So, January 1, 1981, plots at 1981.0. These rules apply regardless of the PA selected. Labels on graphs showing four years or less on the horizontal axis provide month information, while those for longer periods simply show time in decimal years.

Selecting Units of Measurement for Graphs and Tables in EGRET

Data in the primary data frames of EGRET (Daily and Sample) and many of the data frames and matrices that store various summaries of results use International System of Units (SI) units of measurement. There is one possible exception for discharge data in cases not involving water-quality data analysis, and this is described in section "Discharge Data from the U.S. Geological Survey Data Service." The units for stored discharge values are m^3/s, for concentration they are mg/L, for flux they are kilograms per day (kg/d), and for drainage area they are km^2. However, EGRET provides a high degree of flexibility for users to produce output in units that are more customary for the audience being addressed. EGRET is designed so that, in most cases, careful selection of units can avoid having the output require the use of scientific notation, which makes graphs harder to read. The choices of output units are selected in the graph and table functions through the use of an argument, which is qUnit for discharge and fluxUnit for flux. Figure 2 provides a list of the options available for qUnit and fluxUnit.

```
> printqUnitCheatSheet()
The following codes apply to the qUnit list:
1 =  cfs  ( Cubic Feet per Second )
2 =  cms  ( Cubic Meters per Second )
3 =  thousandCfs  ( Thousand Cubic Feet per Second )
4 =  thousandCms  ( Thousand Cubic Meters per Second )

> printFluxUnitCheatSheet()
The following codes apply to the fluxUnit list:
1 =  poundsDay  ( pounds/day )
2 =  tonsDay  ( tons/day )
3 =  kgDay  ( kg/day )
4 =  thousandKgDay  ( thousands of kg/day )
5 =  tonsYear  ( tons/year )
6 =  thousandTonsYear  ( thousands of tons/year )
7 =  millionTonsYear  ( millions of tons/year )
8 =  thousandKgYear  ( thousands of kg/year )
9 =  millionKgYear  ( millions of kg/year )
10 =  billionKgYear  ( billions of kg/year )
11 =  thousandTonsDay  ( thousands of tons/day )
12 =  millionKgDay  ( millions of kg/day )
```

Figure 2. Computer input and output showing the available choices of discharge units and flux units.

Users can always enter the command printqUnitCheatSheet() or printFluxUnitCheatSheet() to print these lists. For each function, the user can select the units either by providing the code number or the name of the units. For example, if the user wanted to express discharge in thousands of ft^3/s, this is done either by specifying qUnit=3 or qUnit=thousandCfs as the argument to the relevant graphic or table function. Default units are always specified for any of these functions.

Flow History Analysis

Flow history analysis in EGRET provides a very simple description of long-term variability and trend in discharge at a given streamgage. In addition to the ordinary year-to-year variations in precipitation, many other factors can lead to long-term changes in discharge, including:

- changes in consumption of water or diversions of water into or out of the watershed,
- changes in groundwater storage in the watershed that influence base flow or storm flow,
- construction and operations or removal of dams and levees,
- changes in land-use patterns and practices (irrigation, urbanization, subsurface drainage, wetland drainage, or changes from deep-rooted natural vegetation to shallow-rooted agricultural crops),
- climate variability (phenomena such as El Niño, Pacific Decadal Oscillation, or Atlantic Multidecadal Oscillation), or
- long-term nonstationarity of climate possibly due to changes in greenhouse gas and particulate concentrations in the atmosphere.

The methods presented here are intended to help the hydrologist determine the nature of the changes or variations that may be taking place in a given watershed, which can be used to pose or test hypotheses about the possible causes of these changes. The analyses are also to help engineers and planners understand changing flow conditions that may be crucial to designing or planning flood-protection measures, wastewater permits, or water availability for off-stream or in-stream uses. These analyses need to consider many different aspects of the discharge record. They may be focused on average discharges, low flows, or high flows. They may also be focused on a full annual perspective, or may be focused on one particular season. Seasonal analyses could be important for certain types of questions. One example could be a concern about the warming in winter months and determining what influence that might have on winter discharge in watersheds that have historically had large snowpacks and winter temperatures well below freezing. Another example would be a concern about decreasing summertime minimum streamflow in a watershed where groundwater storage has significantly declined, potentially diminishing base flow to streams.

Flow history analysis in EGRET uses the daily discharge record for a given streamgage, organizes it according to some PA (such as water year or winter season), and then evaluates for every year in the PA a set of statistics:

1. the minimum 1-day daily mean discharge,
2. the minimum 7-day mean of the daily mean discharges,
3. the minimum 30-day mean of the daily mean discharges,
4. the median of the daily mean discharges,
5. the mean of the daily mean discharges,
6. the maximum 30-day mean of the daily mean discharges,
7. the maximum 7-day mean of the daily mean discharges, and
8. the maximum 1-day daily mean discharge.

After computing annual time series of these eight statistics, flow history analysis also produces a smoothed version of those time series, which emphasizes the broad multiyear variations and changes in the central tendencies of these time series.

With one exception (described below), the discharge statistics computed are all annual values (one value per year) where the year is as defined by the `paStart` and `paLong` parameters. That is, the period over which the statistic is computed starts with the first day of the month designated by `paStart` and runs through all the days in all of the `paLong` months of the PA. So, for example if `paStart` is 12 and `paLong` is 4, the annual values of the statistic will be computed over the period from December 1 through March 31, and that statistic will be associated with the calendar year in which the period ends (the year containing the March 31 date that forms the end of the period). The one exception is the water year (`paStart=10, paLong=12`). In this case, all of the low-flow statistics (1-day minimum, 7-day minimum, and 30-day minimum) are computed for the climate year, which runs from April 1 through March 31. The use of the climate year, rather than the water year, for low-flow statistics is a common practice in hydrology, because it minimizes the probability that individual drought events will span multiple water years, which are by convention bounded by months that are typically low-flow months, and thus are counted twice in the same time series (Riggs, 1982; Gordon and others, 1991).

The Smoothing Method Used in Flow History Analyses

The analysis of long-term variation in discharge characteristics used in this study builds on time-series smoothing methods that were pioneered by Cleveland (1979) and Cleveland and Devlin (1988). It is designed for analysis of long records

(e.g. greater than 50 years duration, although it will work for shorter records) and it performs smoothing on annual statistics relevant to annual low flow, high flow, or annual mean flow. For this discussion define the discharge for year i as Q_i where that discharge can be any one of the eight streamflow statistics named in the preceding section (such a 7-day minimum, mean, or 1-day maximum) for the PA. For any given year i and discharge (Q_i) there is an associated time value (T_i), which is expressed in decimal years, as described in the section "Setting the Period of Analysis for Graphs, Tables, and Analyses in EGRET." Thus, in a record of n years, for any given flow statistic and PA, there is a set of n values of Q_i and T_i which constitute the time series to be smoothed.

The smoothing method is based on locally weighted scatterplot smoothing (lowess) but with particular features that are described below. The purpose of producing the smooth curves is to extract patterns of change that describe variations at time spans of about a decade or more. Such curves are very resistant to the influence of one or two years with extremely high or low flows. Unfortunately, in those cases where the changes are actually quite abrupt, for example, those caused by construction or removal of a dam or initiation of a major new water diversion, the curves depict those changes as if they were gradual.

The variable y_i is the log-transformed value of the flow statistic:

$$y_i = \ln\left(Q_i\right) \tag{1}$$

The logarithm transformation is applied because discharge data typically are highly skewed, approximating a log-normal distribution in many cases. Use of the logarithm transformation results in weighted regressions in which the residuals are more nearly normal and thus, individual extreme values do not exert a large amount of influence on the estimates. This results in a more robust smoothing process. It also means that the smoothed values, denoted $\hat{Q}_i$, are more nearly an approximation of the median of the time series than they are an approximation of the mean (see Helsel and Hirsch (2002), pages 253–260 for a discussion of transformation issues).

In log-space, the smooth curve is defined by a series of n-weighted regressions on the data set. The estimate, $\hat{y}_i$, of y_i is defined as

$$\hat{y}_i = \beta_{0i} + \beta_{1i} \bullet T_i \qquad \text{for } i = 1, n \tag{2}$$

where

β_{0i} is the estimated regression intercept for the regression model fitted for year i, and
β_{1i} is the estimated regression slope for the regression model fitted for year i.

The two regression coefficients, β_{0i} and β_{1i}, are computed from a weighted regression, where the weights are equal to 1 for the observation for the year in which the estimate is being made, and decay to zero at a time separation of h_i years between a given observation and the time of the estimate. The parameter h_i is the half-window width used to complete the weights for the estimate $\hat{y}_i$. The EGRET software provides an option "`edgeAdjust`" which causes the window to become wider for years close to the start or end of the record and narrower in the middle years. Using `edgeAdjust = TRUE` prevents the smoothed curve from having an excessive amount of curvature near the beginning or end of the record. The default option for the relevant EGRET functions is `edgeAdjust = TRUE`.

Define H as the nominal half window width. The default value of H is set to 20 years, but the user can modify it. Without the `edgeAdjust` feature, the actual half window width for year i, denoted as h_i, is always equal to H. When the `edgeAdjust` feature is in effect then the half window width for the estimate for year i is:

$$h_i = \max\left\{H, 2H - \min\left(T_i - T_1, T_n - T_i\right)\right\} \tag{3}$$

where T_1 is the time value for the first year in the record, T_n is the time value for the last year in the record, and T_i is the time value of the year for which the estimate is being made. Note that when T_i is equal to the time of the first or last value, $h_i = 2H$.

The specific weights are computed with the tricube weight function. The weight for the j^{th} streamflow value in the computation of the smoothed value for the i^{th} year is:

where,

$$w_{i,j} = \begin{cases} \left(1 - \left(|\, d_{i,j} \,| / h_i\right)^3\right)^3 & if \left|d_{i,j}\right| \le h_i \\ 0 & if \left|d_{i,j}\right| \ge h_i \end{cases} \tag{4}$$

$$d_{i,j} = T_i - T_j \tag{5}$$

Figure 3 shows the shape of the weight function for three cases using the `edgeAdjust` option and the default value of *H,* which is 20 years. The first case, which applies when the year being estimated (T_i) is far from either end of the record, shows that all data values that are between zero and 10 years from the estimated year have weights that are at least 67% as large as the largest weight. The other two examples show cases where the year being estimated is near or at the end of the record. In each of these cases, higher weights are extended farther from the year being estimated, in order to provide more stability since there are few or even no data points to balance the data set. The default "half-window width" where $H = 20$ years, was selected by visual examination of graphics for many alternatives for many different discharge records. It was selected to be as narrow as possible, such that individual year-to-year oscillations are fully damped out. Although 20 years is the default value for the half-window width used in EGRET, it can be modified by the user, using the `window` argument in the function `setPA`.

The final step in producing the set of smoothed annual values is the retransformation:

$$\hat{Q}_i = \exp\left(\hat{y}_i\right) \tag{6}$$

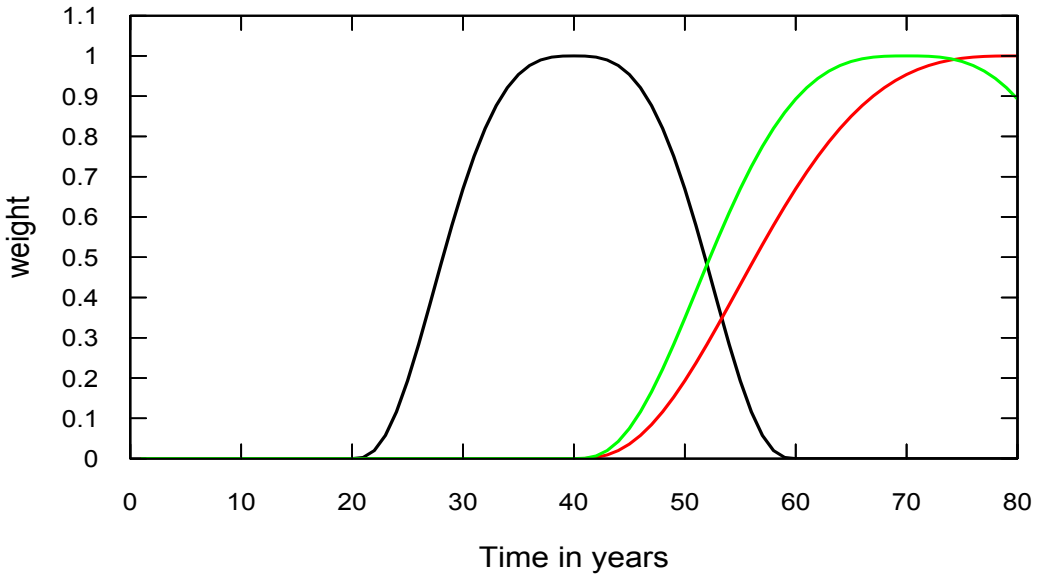


Figure 3. Graph showing values of the tricube weight function used in weighted regressions for smoothing discharge time series of 80 years duration. Black curve is the weight function for estimates for year 40, green is for estimates for year 70, and red is for year 80. The half-window width (*H*) is set to its default value of 20 years with `edgeAdjust = TRUE`.

This retransformation is designed to produce a smooth representation of the median of the distribution over time for the statistic being plotted.

The next section shows the EGRET commands needed to run this type of analysis and provides examples of graphical and tabular outputs available.

EGRET Functions for Flow History Analysis

The set of commands needed to conduct the analyses described in the proceeding section is defined here. For completeness, the commands here include the data retrieval steps described more fully in the section of this report "Daily mean Discharge Data For Use in EGRET." This example will use the discharge record for the Spokane River at Spokane, Washington, USGS streamgage 12422500, because the record is very long (starting 1891–04–01), is complete to the present, and has shown substantial change, particularly in terms of declining low-flow conditions, due to a long history of depletion of water from the Spokane Valley-Rathdrum Prairie Aquifer. Each command is shown here with an explanation of its purpose immediately after it.

```
siteNumber <- "12422500"
```

This specifies that the streamgage of interest is USGS number 12422500.

```
param <- "00060"
```

This specifies that the parameter of interest is daily discharge (00060).

```
Daily <- readNWISDaily(siteNumber, param, startDate = "", endDate = "2014-06-30")
```

This retrieves the discharge data set from USGS data services and forms it into the Daily data frame, which includes daily discharge as well as 7-day and 30-day average discharge for each date in the record.

```
INFO <- readNWISInfo(siteNumber, param)
```

This initiates an interactive process of retrieving the metadata for the site and parameter. It provides some of that information to the user and prompts the user for various types of text information for labeling of files and graphics. All of the metadata, including the user responses to prompts, are stored in the `INFO` data frame. After the `INFO` and `Daily` data frames have been created, they need to be organized into the list called `eList`, using command:

```
eList <- as.egret(INFO,Daily)
```

The Function `setPA`

This function is required for setting up the PA and the window width to be used in the analysis. In the default case, the function sets up the period of analysis as the water year and sets the nominal half-window width H to 20 years. The function performs no computations, but simply augments the metadata stored in the `INFO` data frame (stored in `eList`) with the `paStart`, `paLong` and `window` values (where H = `window`). In the default case, the command is:

```
eList <- setPA(eList)
```

If the user were selecting other values, for example, a PA that covers the months December through February and a half window width of 15 years, the call would be:

```
eList <- setPA(eList, paStart = 12, paLong = 3, window = 15)
```

The Function `makeAnnualSeries`

This is a function that the user doesn't call directly, but it is central to the computations of virtually every one of the Flow History functions and, as such it is worthwhile to document it here to provide background information to the user. This function creates a matrix called `annualSeries` to store the annual series of discharge statistics. It computes and stores those statistic values in `annualSeries`, and then computes the smoothed estimates of these statistics and stores them in `annualSeries`.

The dimensions of `annualSeries` are (3,8,n) where n represents the number of years for which one or more of the annual discharge statistics can be calculated. In the Spokane River example shown above, `annualSeries` has dimensions (3,8,125). For the first dimension: 1 is for storing the mean value of `decYear`, the decimal time value, for all of the daily discharge values used to compute the statistic of interest; 2 is the actual discharge statistic for that particular year; and 3 is the smoothed value of that discharge statistic for that particular year. The second dimension is the index of the discharge statistic. The index of these discharge statistics is known as `istat`. The `istat` values for the eight discharge statistics are defined in table 5.

Table 5. Definitions of the eight discharge statistics computed in EGRET.—Continued

istat	Statistic name
1	Annual minimum1-day daily mean discharge.
2	Annual minimum 7-day mean of the daily mean discharges.
3	Annual minimum 30-day mean of the daily mean discharges.

Table 5. Definitions of the eight discharge statistics computed in EGRET.—Continued

istat	Statistic name
4	Annual median of the daily mean discharges.
5	Annual mean of the daily mean discharges.
6	Annual maximum 30-day mean of the daily mean discharges.
7	Annual maximum 7-day mean of the daily mean discharges
8	Annual maximum 1-day daily mean discharges.

The third dimension of the `annualSeries` matrix is an index of years, from 1 through n. There can be NA values for some of these statistics if the amount of data available for computing the statistic is less than 90 percent of the number of days that would exist in a complete record for that period. Thus, the first year that will have a value is the first year in which the data set for the PA is 90 percent or more complete, and similarly the last year that will have a value is the last year in which the data set for the PA is 90 percent or more complete. For example, for a water year or calendar year, there must be more than 328 days of available data in order for the statistic to be computed for that year. Note that because of the way that the 7-day or 30-day statistics (`istat` = 2, 3, 6, and 7) are computed, the set of days that define the annual discharge statistic can sometimes be centered on a date that slightly precedes the specified PA of the specified year.

In the example shown here, the annual 7-day minimum discharge for the first year of the record is stored as `annualSeries[2,2,1]`. Its smoothed value is stored as `annualSeries[3,2,1]`. The mean decimal year value for the dates used to compute this discharge statistic is stored as `annualSeries[1,2,1]`.

Plotting the Results for a Single Discharge Statistic by Using plotFlowSingle

The actual values of a given discharge statistic and their smoothed values can be plotted using the function `plotFlowSingle`. The specific discharge statistic (`istat`) must be specified when invoking this function. Many other arguments can be specified (see `?plotFlowSingle` or appendix 2), yet the only one that is mentioned here is the selection of discharge units to be used. Discharge units are specified by the argument `qUnit`. In this example, the unit m^3/s is specified, (`qUnit = 2`). The default is `qUnit = 1`, which is ft^3/s. The function call for annual 7-day minimum discharge, in units of m^3/s is:

```
plotFlowSingle(eList, istat = 2, qUnit = 2)
```

The resulting graphic is shown in figure 4.

Printing Results for a Single Discharge Statistic by Using printSeries and tableFlowChange

To generate a table suitable for printing or for input to some other analysis, the output from `plotFlowSingle` can be displayed by using the function `printSeries`. The only argument that must be specified for this function is `istat`; all others can be their default values. Producing a table of numbers that corresponds to figure 4, but with the discharge values reported as runoff in mm/day, can be done with the following command:

```
SpokaneSeries <- printSeries(eList, istat = 2, runoff = TRUE)
```

This command 1) creates a data frame, in this case called `SpokaneSeries`, that contains the tabular information, and 2) prints the information to the console. The printed output is shown in figure 5.

By use of the data frame called `SpokaneSeries` created in this example, these results can be written to a file, which can then be used as input to some other computer application including input to a spreadsheet or a word processing application for use in producing a table suitable for publication. Details on how to make such conversions are provided in appendix 2, section 12. In addition, the results shown in figure 4 can also be expressed in terms of the amount of change estimated to have taken place between any two years in the smoothed time series. This is done with the function `tableFlowChange`. The function describes these changes between selected pairs of years in four different ways. For a comparison between two times, T_i and T_j (expressed in decimal years), where the smoothed values of discharge at those times are denoted $\hat{Q}_i$ and $\hat{Q}_j$, `tableFlowChange` will express the change as:

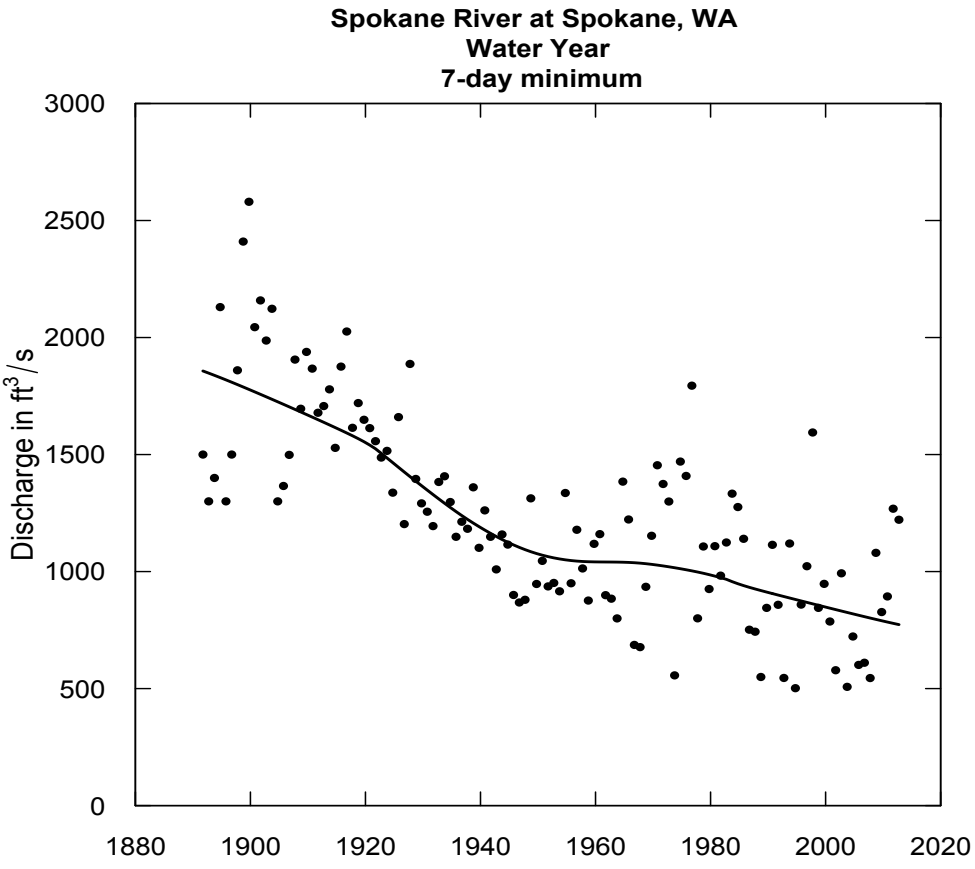


Figure 4. Plot of the 7-day minimum discharge by year (dots) and smoothed estimates (curve) for the Spokane River at Spokane, Washington.

1. a change between the first and last year of the pair, expressed in the flow units selected, $\hat{Q}_j - \hat{Q}_i$,
2. a change between the first and last year of the pair, expressed as a percentage of the value in the first year, $(\hat{Q}_j - \hat{Q}_i) \bullet 100 / \hat{Q}_i$,
3. a slope between the first and last year of the pair, expressed in terms of the flow units per year, $\left(\hat{Q}_j - \hat{Q}_i\right) / \left(T_j - T_i\right)$, and
4. a slope between the first and last year of the pair, expressed as a percentage change per year (a percentage based on the value in the first year), $\left(\hat{Q}_j - \hat{Q}_i\right) \bullet 100 / \left(\hat{Q}_i \bullet \left(T_j - T_i\right)\right)$.

Five arguments are needed by `tableFlowChange`. The first is `eList`, which is the list containing the `INFO` and `Daily` data frames. The second is `istat,` which determines the flow statistic to be analyzed. This argument has been introduced earlier, and it has eight possible values (table 5). The third is `qUnit`. The default value is `qUnit` = 1, which corresponds to ft^3/s, but all four options for qUnit values listed in figure 2 are possible. The fourth is `runoff`, for which the default value is `FALSE`. If `runoff` is `TRUE`, then the results are in units of mm/day. The final argument is `yearPoints`. The object `yearPoints` is a vector of integer numbers that represents the full set of years for which the user wants to make comparisons. This is best explained by an example. If one wanted to examine the changes in discharge from 1950 to 1980 and 1980 to 2010, then `yearPoints` would be specified as:

```
yearPoints <- c(1950, 1980, 2010)
```

The `yearpoints` argument above indicates that the full set of comparisons (changes or slopes) that is made includes all possible ordered pairs of these years, which are 1950–80, 1950–2010, and 1980–2010. In the example of the Spokane River 7-day minimum flow, one might choose comparisons in blocks of 40 years each. The commands would be:

```
yearPoints <- c(1892,1932,1972,2012)
tableFlowChange(eList, istat = 2, qUnit = 2, yearPoints=yearPoints)
```

```
Spokane River at Spokane, WA
 Water Year
    7-day minimum
    runoff in mm/day
   year   annual   smoothed
           value    value

   1892    0.330    0.379
   1893    0.286    0.379
   1894    0.308    0.380
   1895    0.469    0.381
   1896    0.286    0.381
   1897    0.330    0.381
   1898    0.410    0.382
   1899    0.531    0.382
   1900    0.568    0.382
   1901    0.450    0.383
   1902    0.475    0.383
. . . output edited. . . .
   2003    0.219    0.184
   2004    0.112    0.184
   2005    0.159    0.184
   2006    0.132    0.184
   2007    0.134    0.184
   2008    0.120    0.185
   2009    0.238    0.185
   2010    0.182    0.185
   2011    0.197    0.185
   2012    0.279    0.186
   2013    0.269    0.186
   2014    0.223    0.186
```

Figure 5. Output of the annual 7-day minimum discharge and smoothed values of annual 7-day minimum discharge for the Spokane River at Spokane, Washington.

or it could be done in a single command line as:

```
tableFlowChange(eList, istat = 2, qUnit = 2, yearPoints = c(1892,1932,1972,2012))
```

The output is shown in figure 6.

Plotting Changes in Variability by Using plotSDLogQ

This graphic is designed to indicate if changes are taking place in the overall variability of the daily discharge record, without regard to the question of whether the central tendency is changing over time. The function `plotSDLogQ` produces a graphic of the running standard deviation of the log of daily mean discharge. By using the standard deviation of the log discharge, the statistic is dimensionless. It is a measure of relative variability. If, for example, the probability distribution of daily mean discharge were to have trended upwards (or downwards) over time, but had done so in a manner that all quantiles of the distribution had increased by the same percentage amount, then we would expect this graphic to show a horizontal line. If, on the other hand, the change in the probability distribution were such that there was a greater percentage change in the high

```
  Spokane River at Spokane, WA
  Water Year
   7-day minimum

            Streamflow Trends
  time span          change        slope       change        slope
                       cms        cms/yr         %            %/yr
1892  to  1932          -12        -0.29          -24          -0.6
1892  to  1972          -19        -0.23          -39         -0.48
1892  to  2012          -25        -0.21          -51         -0.42
1932  to  1972           -7        -0.18          -19         -0.48
1932  to  2012          -13        -0.16          -35         -0.44
1972  to  2012         -6.1        -0.15          -20         -0.51
```

Figure 6. Output from `tableFlowChange` for the Spokane River at Spokane, Washington.

end and (or) low end of the distribution, compared to the percentage change in the middle portion of the distribution, then this curve would slope upwards over time. It is much like a graph of a moving coefficient of variation, but it has sample properties that make it a smoother measure of the changing variability in the data. For example, this graph can be useful and simple way of providing empirical evidence for hypotheses exploring the idea that increasing urbanization or increasing greenhouse gas concentrations in the atmosphere are bringing about changes in hydrologic variability. The function is called with the command:

```
plotSDLogQ(eList)
```

The computation is made by using a moving window over which the standard deviation is computed. The default width of this window is 15 years, and the user can select a different width by using the argument window. So if the user wanted to use 20 years, the command is:

```
plotSDLogQ(eList, window = 20)
```

If the 15 year default window is used, the computations begin with a window centered on a date that is 7.5 years from the start of the record. The standard deviation of the natural logarithm of the daily mean discharges is computed for all the days from 7.5 years before to 7.5 years after that center date, and the result is what is plotted on the graph, for that center date. Note that the "window" argument to `plotSDLogQ` specifies a nonweighted rectangular window over which the standard deviation is computed, in contrast to the "window" argument to `setPA`, which specifies a weighted smoothing window for computing other statistics. The computation then moves on to a new center date that is a tenth of a year later (36.5 days later) and repeats as many times as necessary until it reaches a center date that is 7.5 years before the end of the record. The user can limit the span of the calculation with the use of two arguments, `yearStart` and `yearEnd`, expressed in decimal calendar years. The defaults for these arguments are the start and end of the available record in the `Daily` data frame. For an example of this plot (fig. 7), the data set used is the discharge record from the Colorado River at Lees Ferry, Arizona. This case is an interesting one because it shows such a strong trend towards decreasing variability, a result of the history of increasing water management through reservoir storage, which decreases the size of the highest discharges and increases the size of the lowest discharges.

If the user is interested in changes in variability in a particular season, the desired PA is specified prior to running, `plotSDLogQ` by using the `setPA` function. For example, to consider variability in the months of June, July, and August, the commands are:

```
eList <- setPA(eList, paStart = 6, paLong = 3)
plotSDLogQ(eList)
```

The resulting graph is appropriately labeled to indicate the PA used.

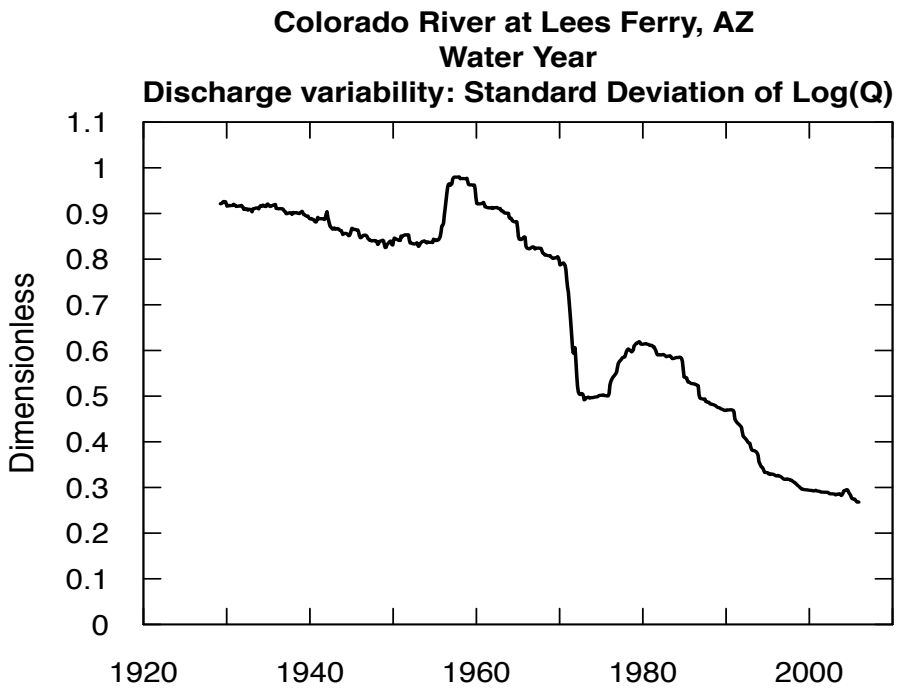


Figure 7. Plot of the standard deviation of Log(Q) over time, Colorado River at Lees Ferry, Arizona.

Creating Graphics for Plotting the Discharge Record by Using plotQTimeDaily

Plotting the complete discharge record is a common task and the function `plotQTimeDaily` makes that possible. The function can be called with all of its arguments set to their default values, and it produces a suitable graphic. Figure 8 presents an example of such a plot for the Big Sioux River at Akron, Iowa, for 1941–2011. The command in this case is:

```
plotQTimeDaily(eList, lwd = 1,qUnit = 2)
```

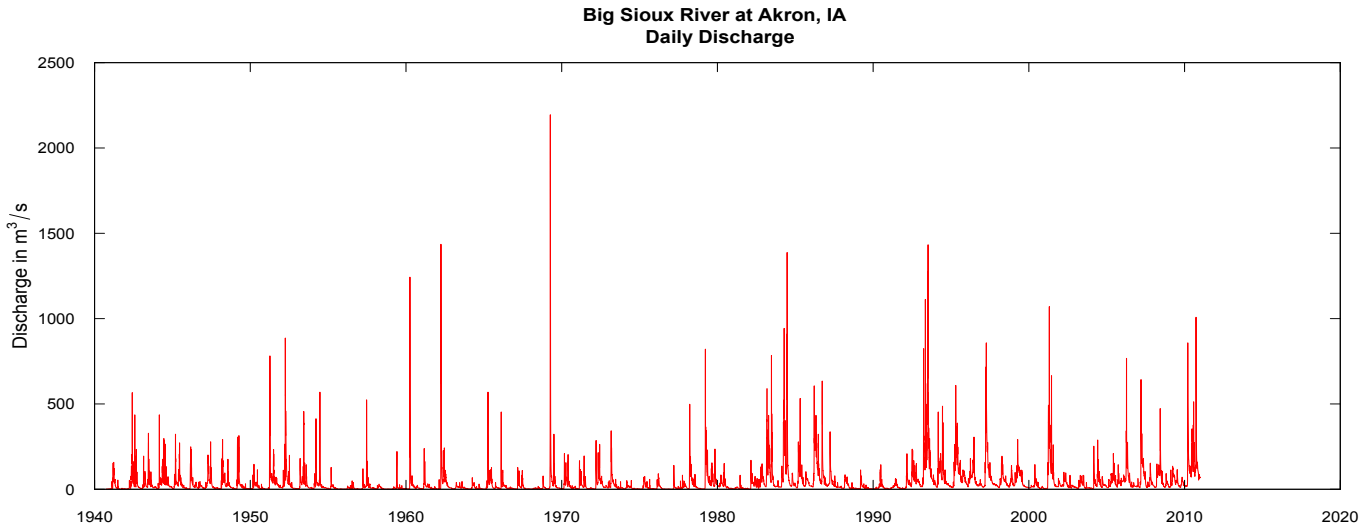


Figure 8. Output from `plotQTimeDaily` of a complete daily discharge record for the Big Sioux RIver at Akron, Iowa.

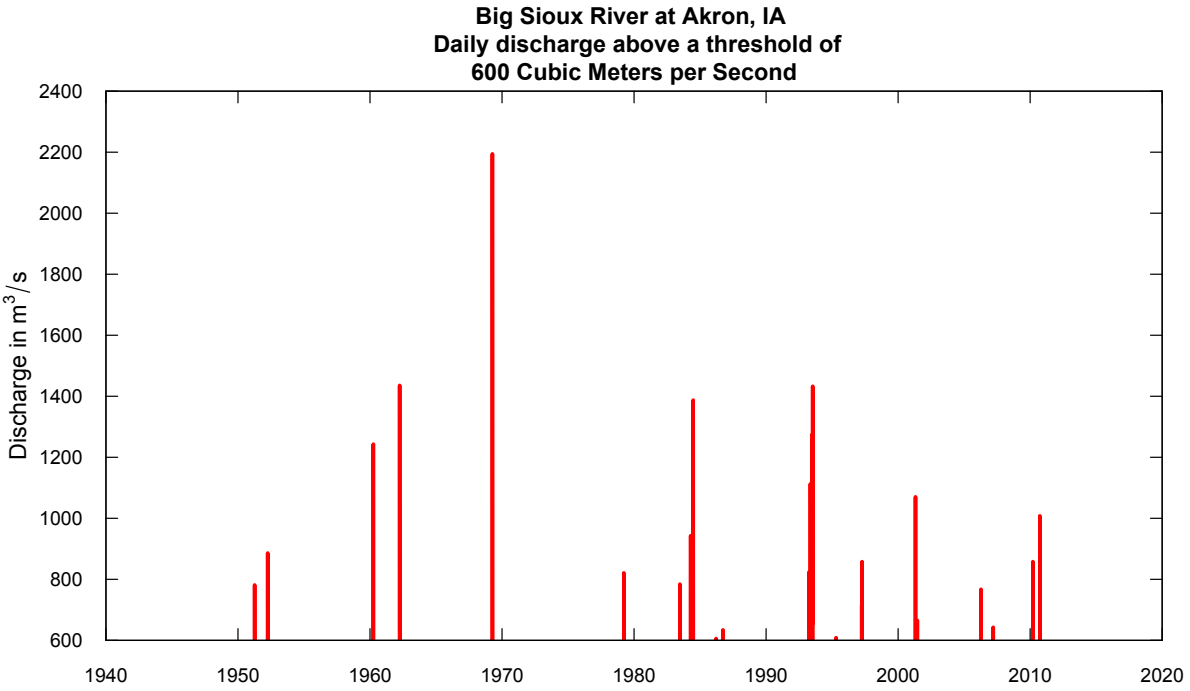


Figure 9. Output from `plotQTimeDaily` for discharge data for Big Sioux River at Akron Iowa, showing only discharges greater than 600 cubic meters per second.

The specific arguments called are `lwd = 1` for setting the line width narrower than the default (which is `lwd = 3`) and `qUnit = 2` (to select m^3/s as the units for the graph). The color red is the default, which prevents segments of the discharge record from being confused with the black tick marks.

Another use for this function is to plot only those periods in which discharge is higher than some threshold value. This can be useful for discussions that relate to possible changes in the magnitude and (or) frequency of high discharge events. By using the same record, with a threshold of 600 m^3/s, the command `plotQTimeDaily(eList, qLower = 600, lwd = 3)` produces the graph in figure 9.

A graphic of this type is useful for analyzing the magnitude and frequency of high discharge events. It can be very helpful for illustrating the persistence of high discharge events (the tendency for high flows to occur in groups rather than a random pattern) and for revealing patterns of changing magnitude and (or) frequency of high discharge.

Creating Multipanel Graphics for Flow History

There are three specific multipanel graphics functions for depicting flow history information. One of these is `plotFour`, and it is a combination of three panels produced by `plotFlowSingle` and one panel produced by `plotSDLogQ`. The three panels from `plotFlowSingle` are those for the 1-day maximum, the mean, and the 7-day minimum discharges. These panels can be done for any PA, depending on the questions of interest. Although `plotFlowSingle` and `plotSDLogQ` each allow the user to set a maximum value for the vertical axis, this function, and others similar to it, set the value automatically so that the scales for each panel extend to a value slightly larger than the maximum value in the specific data set being plotted in that panel. Users should recognize that the annual 1-day maximum is not the same as the annual peak discharge. The annual peak discharge is intended to represent an instantaneous maximum discharge value for the year. The 1-day maximum will always be a smaller value. On large rivers where discharge changes slowly, there may be very little difference between the 1-day maximum and the annual peak, and the peaks typically will be very highly correlated with each other. On small streams where discharge can rise from a very low flow value to an annual maximum within a single day, there can be very large differences between these values and the correlation between the two time series may be very low. Figure 10 is an example produced for the Big Sioux River.

This type of graphic can be particularly useful for exploring changes that may be focused on one particular part of the year. The following example is for the very long discharge record of the Merced River at Happy Isles Bridge, which is high in the Sierra Nevada Mountains in Yosemite National Park, California. The warming conditions of the past several decades are having an impact on wintertime flow conditions, and this is illustrated with this example where the PA is the 2 months January

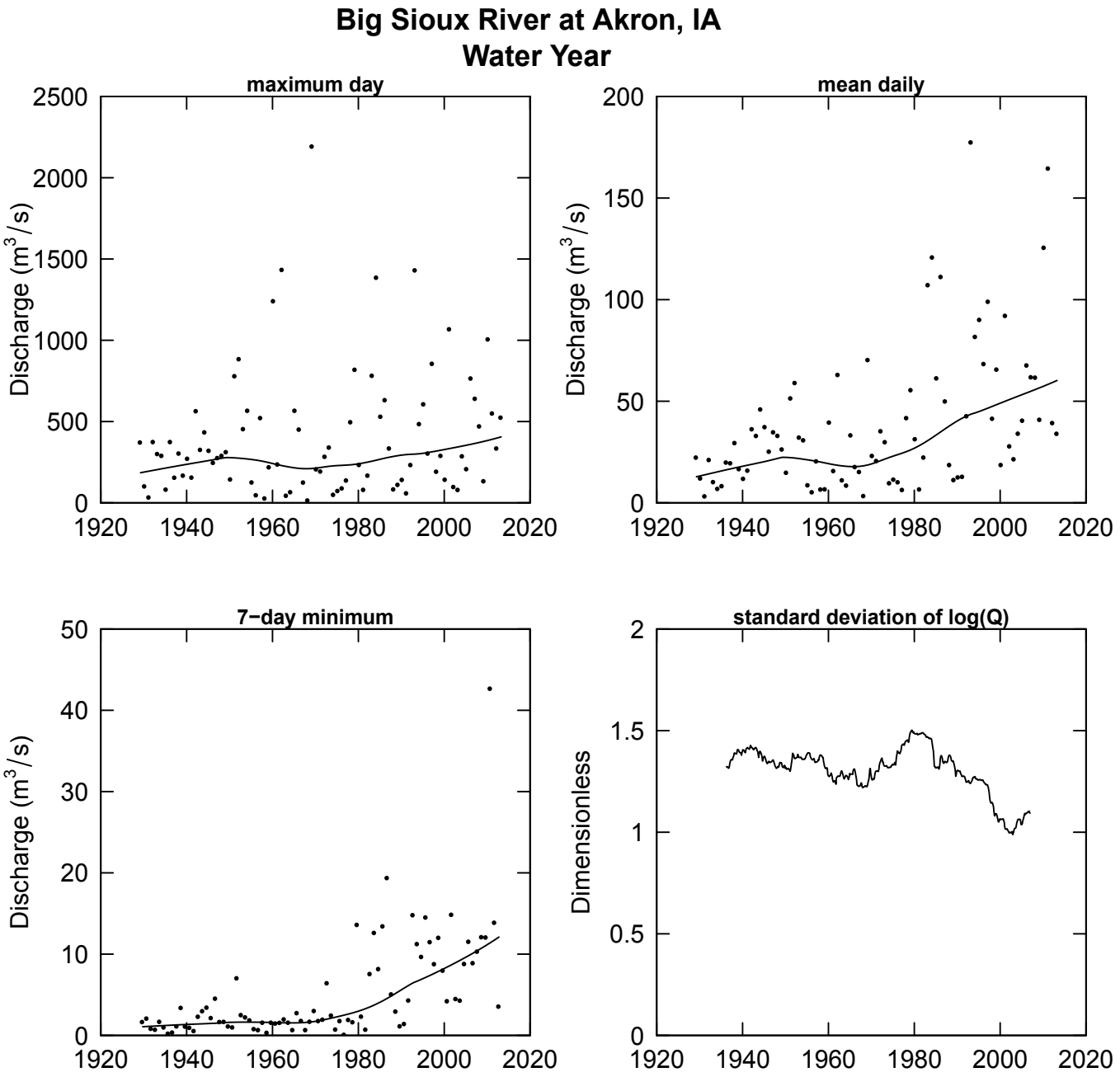


Figure 10. Example of graphics produced by the `plotFour` function for the Big Sioux River at Akron, Iowa.

and February. The commands to produce figure 11 are this combination (after the `Daily` and `INFO` data frames and `eList` are created):

```
eList <- setPA(eList, paStart = 1, paLong = 2)

plotFour(eList, qUnit = 2, window = 21)
```

The next function, `plotFourStats`, is nearly identical to `plotFour`; the only difference is that the standard deviation of log(Q) is not part of `plotFourStats`, and it is replaced by the median discharge. The reason such a similar function is available is that the standard deviation of log(Q) is somewhat unconventional and some users may find it difficult to explain to their audience, and `plotFourStats` provides a more conventional alternative. Users who have some knowledge of R

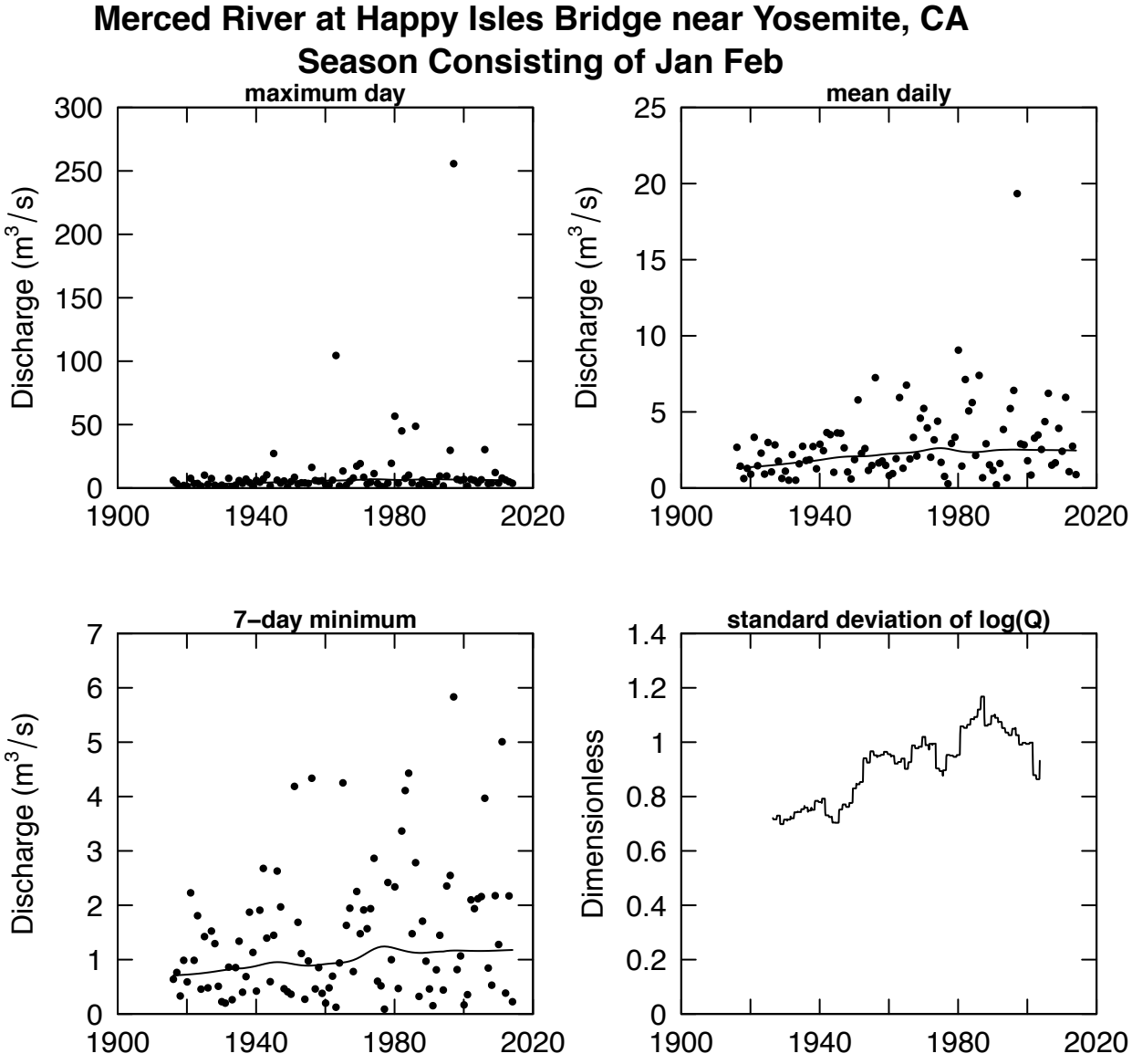


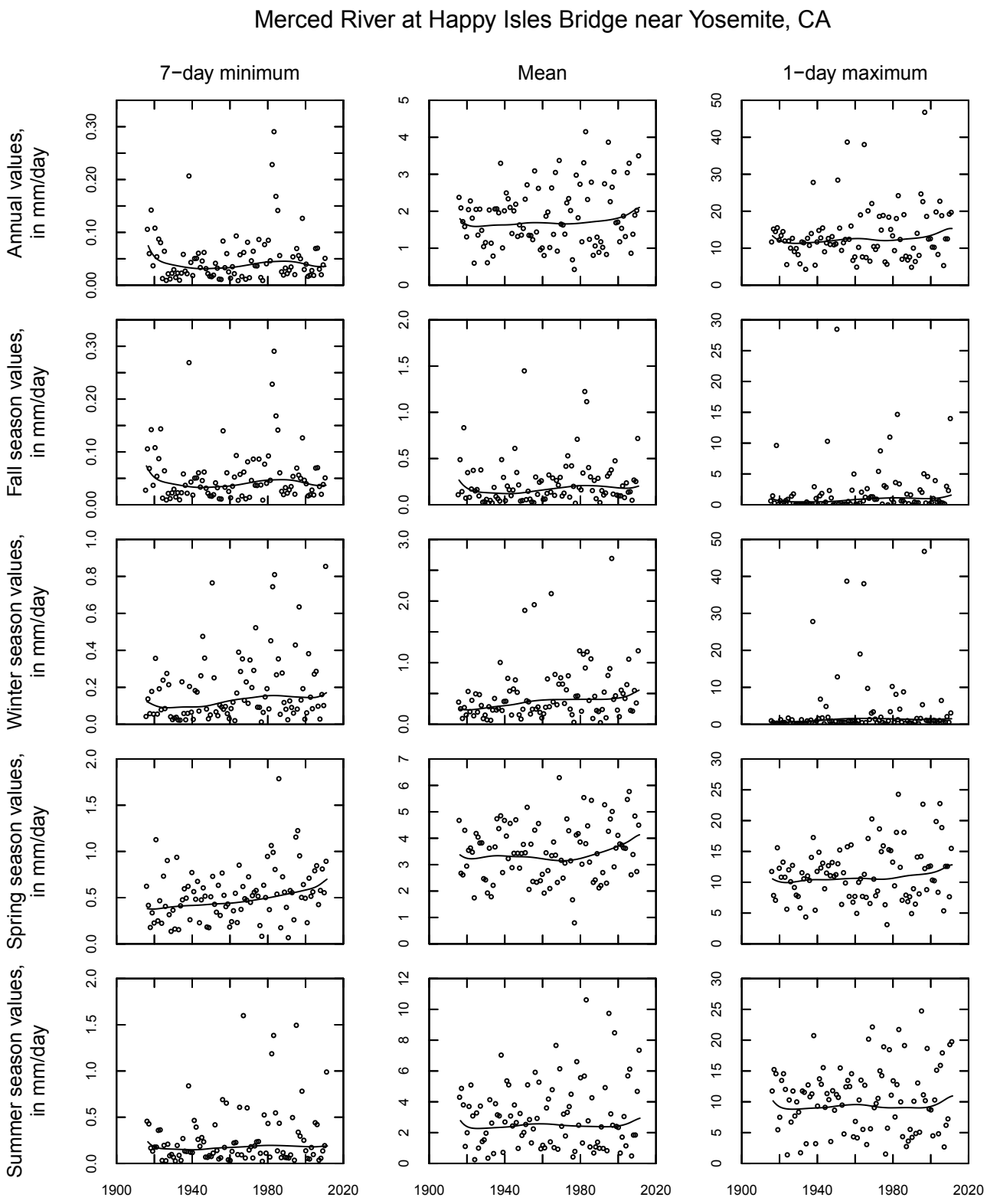
Figure 11. Output produced by the `plotFour` function for the Merced River at Happy Isles Bridge near Yosemite, California.

programming can easily adapt these functions to plot other combinations of the statistics calculated by selecting different values of `istat` into the calls to `plotFlowSingle` that are included in the code for `plotFour`.

Finally, there is the much more elaborate graphic, `plot15`. It produces a set of 15 plots (all based on `plotFlowSingle`). They include the 1-day maximum, mean, and 7-day minimum for five different PAs. These PAs are the water year, Fall (September, October, November), Winter (December, January February), Spring (March, April, May) and Summer (June, July, August). Because of its complexity and detail, this figure is best produced as output to a file, but it can be plotted to the computer screen. Before plotting it to the computer screen, the user should set up a graphics window first to accommodate it (a window with a height that is greater than its width). The commands for producing it as a Portable Document Format (PDF) file are:

```
pdf("plot15.pdf",height=10,width=8)
plot15(eList, yearStart=1900,yearEnd=2012)
dev.off()
```

Figure 12 shows the results of these commands for the Merced River discharge data set.



Streamflow statistics (circles) in units of millimeters per day, annual values and seasonal values
Fall (Sept., Oct., and Nov.), Winter (Dec., Jan., and Feb.), Spring (Mar., Apr., and May), and Summer (June, July, and Aug.)
and locally weighted scatterplot smooth (solid curve) for Merced River at Happy Isles Bridge near Yosemite, CA for 1900 − 2012.

Figure 12. Output produced by the `plot15` function for the Merced River at Happy Isles Bridge near Yosemite, California.

This graphic can be used to provide a quick overview of changes in streamflow for a large number of rivers within a region. An example of its use in that manner is in the appendix to Rice and Hirsch (2012).

Summarizing Water-Quality Data (without Using WRTDS)

The EGRET software includes several functions that produce summary information about water-quality data and the associated discharge data. First, the summary information provides a quick means of identifying problems with the data set by showing observations that seem highly unusual and should be investigated before proceeding with the analysis. If these values can be shown to be clearly in error, they should be deleted or edited so that they are correct. If they appear to be correct, but are highly unusual, their influence on the results of subsequent analysis should be evaluated. Steps that can be taken to remove or edit an observation in the data set and also to explore sensitivity to extreme observations are discussed in the section "Editing Data Sets."

Second, the summary information provides insight into important properties of the data set. For example, it might reveal the nature of the relation between concentration and discharge, the presence and nature of seasonality in the data, or the presence of gradual or abrupt trends in the data. It may also show temporal patterns in the data collection such as long gaps in the record, changes in the frequency of sampling, or very limited sampling during certain seasons. The existence of long gaps (more than about two years) can be important to the interpretation of WRTDS results, and a specific function (`blankTime`) is provided to deal with the issues of long data gaps. In cases where there is little or no sampling during certain seasons (most commonly the winter season), the user may want to restrict the WRTDS results that are presented by setting the PA to the sampled months (for example, by confining the results to April through November by using `paStart = 4, paLong = 8`).

Third, changes in laboratory reporting practices may be revealed, such as shifts in the reporting levels of censored data or changes in rounding practices; for example, values for some period of time are reported as 0.1, 0.2, 0.3, …. but in later years, they are reported with many more significant figures such as 0.137, 0.218, 0.307. When this happens, the user may want to consider these more rounded values to be interval censored (so that 0.1 might be represented as a range of 0.05 to 0.015). The section "Editing Data Sets" provides suggestions for doing this type of data recoding.

The types of plots presented in this section may be highly useful for confirming various findings in WRTDS results. Many audiences, including the data analyst, may view the rather complex smoothing approach used in WRTDS with skepticism. Confirmation of findings by using much simpler graphical means can provide useful confirmation of the WRTDS inferences and suggest further analysis that should be considered.

The following sections describe and show examples of the functions for summarizing water-quality data. They are shown here with most arguments at their default levels and only a few crucial arguments used. The graphic functions described in this section all have the ability to produce plots for data from all times of year or to produce plots that are restricted to a specific PA (see section above "Setting the period of analysis for graphs tables, and analyses in EGRET"). Restricting the graphics to a particular PA is accomplished by using the `setPA` function. If, for example, we wanted a graphic specific to the months December, January, and February, we would enter the command `eList <- setPA(eList, paStart=12, paLong=3)` prior to the specific plotting command. The graphic resulting from that plotting command will state what months are included. If a previous plot used a PA that was not 12 months in length and the next plot was intended to cover the full year, then the command `eList <- setPA(eList)` must be used to reset the PA to the water year (because the defaults for `setPA` include `paStart=10` and `paLong=12`). For more complete descriptions of all of the flexibility of these functions, users should refer to appendix 2.

plotConcTime

This function produces a time series graph of the constituent concentration values as a function of time. In its simplest form the command is:

```
plotConcTime(eList)
```

This produces a graphic of all of the concentration values versus time for the entire period of record (every row of the `Sample` data frame). The convention for showing a censored value is the use of a line segment, rather than a simple point, for these observations. Table 6 lists the important arguments for this function that can be used to modify the y-axis scale or cause the plot to cover a selected subset of the full data set.

Table 6. Important arguments for the function plotConcTime.

Arguments	Purpose and options	Default
eList	Specifies the named list that contains the INFO and Sample data frames that will be used by this function.	No default, name of list must be stated.
logScale	If TRUE, the function plots concentrations on a log scale. If FALSE, the function plots concentration on an arithmetic scale, with minimum of zero.	logScale = FALSE
concMax	Specifies a maximum value for the vertical scale, a concentration in mg/L. The default allows the maximum value to be set automatically based on the data. The concMax argument can be used to standardize a set of graphics through a given report.	concMax = NA
concMin	Only used when logScale=TRUE. Sets the minimum value on the vertical axis. The minimum value must be greater than zero. It is a concentration in mg/L.	concMin=NA
qLower	Sets a lower bound on the discharge values for the set of sample values shown in the figure. It is expressed in the discharge units selected by user. See qUnit argument below.	qLower = NA (this is equivalent to a lower bound of zero)
qUpper	Sets an upper bound on the discharge values for the set of sample values shown in the figure. It is expressed in the discharge units selected by the user.	qUpper = NA (this is equivalent to having no upper bound)
qUnit	Selects the units for discharge values used in qLower and qUpper.	qUnit = 2, which is m^3/s

To understand the way that the various features of this function can be used, we will consider the case of nitrate trends in the Choptank River on the eastern shore of Chesapeake Bay. This is a largely agricultural watershed, with highly permeable soils and shallow aquifers. Increasing concentrations of nitrate in groundwater over the past few decades have been documented in this region (Debrewer and others, 2008) The question to be considered is what kinds of changes have taken place in nitrate in the surface water of this area. The data set is very rich; it is, in fact, the type of data set for which WRTDS was designed. Figure 13 is the graphic produced by the command:

```
plotConcTime(eList)
```

The general idea that concentrations have trended upwards over the period from about 1980 to 2012 is clear at a glance. It is complicated by the change in reporting rules that caused the early years of the record to have the reported values that were greater than 1 mg/L to be rounded to the nearest tenth of 1 mg/L, and after about 1996, this rounding was eliminated. The plotConcTime function allows for more specific exploration of the nature of changes in this system. For example, there may be a strong interest in conditions in a particular part of the year and the changes that are taking place at different discharge conditions. The user can isolate a season, in this case the months of April, May, and June, by setting the PA by using the setPA function. Figure 14 shows the results from the two commands

```
eList <- setPA(eList, paStart = 4, paLong = 3)

plotConcTime(eList)
```

The upward trend continues to be evident when only this season is considered, but it may also be of interest to know if the trend is specifically focused in a particular range of discharge values during this season. This can be done by using the qLower and qUpper arguments, which cause plotConcTime to selectively plot only concentrations on days when the daily mean discharge was greater than qLower, or days when daily mean discharge was less than qUpper, or if both arguments are used, only those data collected on days with discharge in the range between qLower and qUpper. The values of qLower and qUpper are specified in units selected by the user and specified in the argument qUnit. If qUnit is not specified, then the default qUnit = 2 is used, which causes the units to be m^3/s. The full set of options for qUnit is shown in table 2. The following is an example of the use of this capability (it is assumed here that the PA has already been set by setPA; if in doubt, the user can verify it by giving the command eList$INFO or just using the setPA function again). The command used to produce the desired plot is:

```
plotConcTime(eList, qUnit=1,qUpper=165,qLower=34)
```

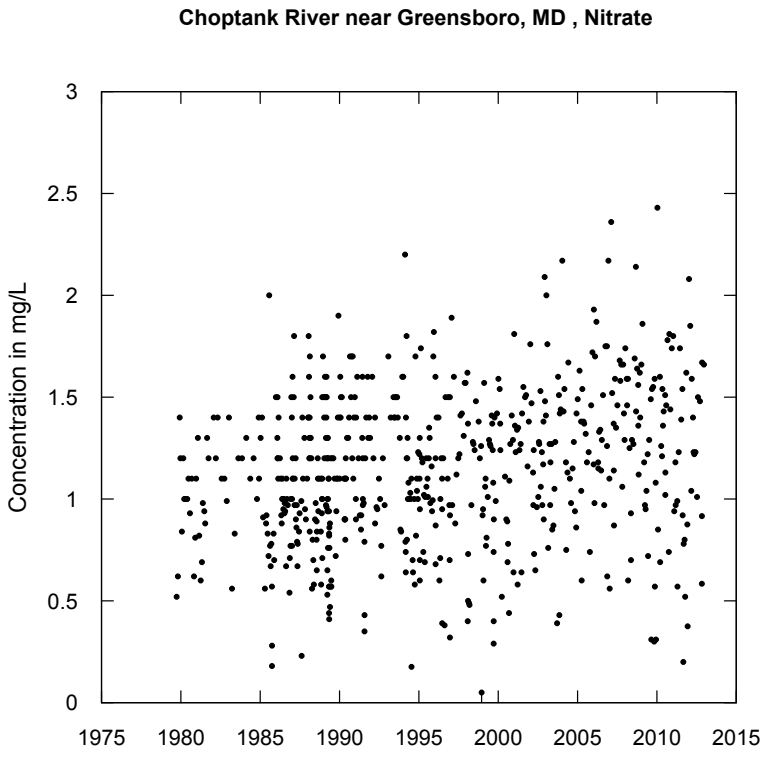


Figure 13. Plot produced by the `plotConcTime` function showing nitrate concentrations over time for the Choptank River near Greensboro, Maryland.

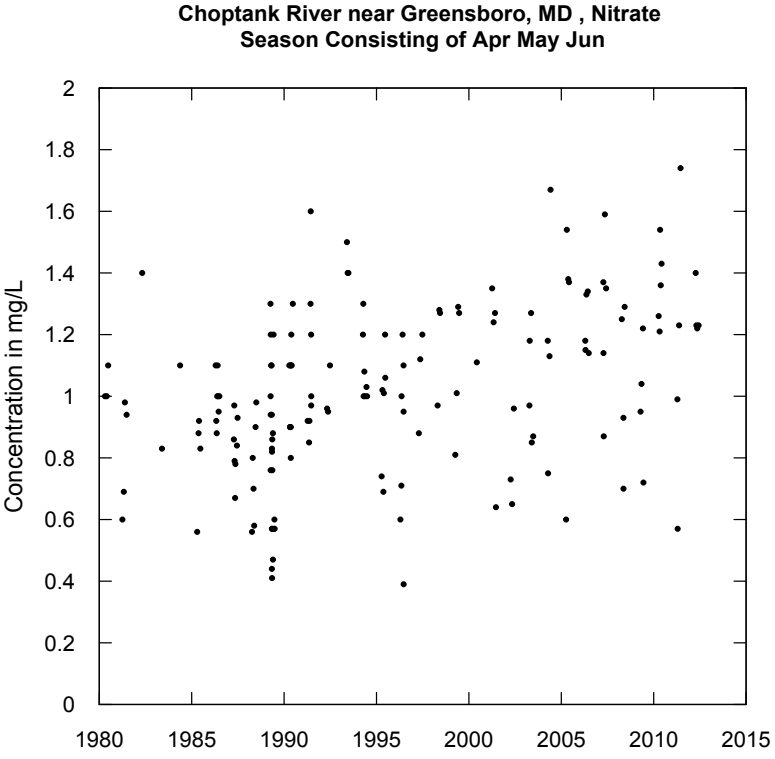


Figure 14. Plot produced by the `plotConcTime` function showing nitrate concentrations over time during April, May, and June for the Choptank River near Greensboro, Maryland.

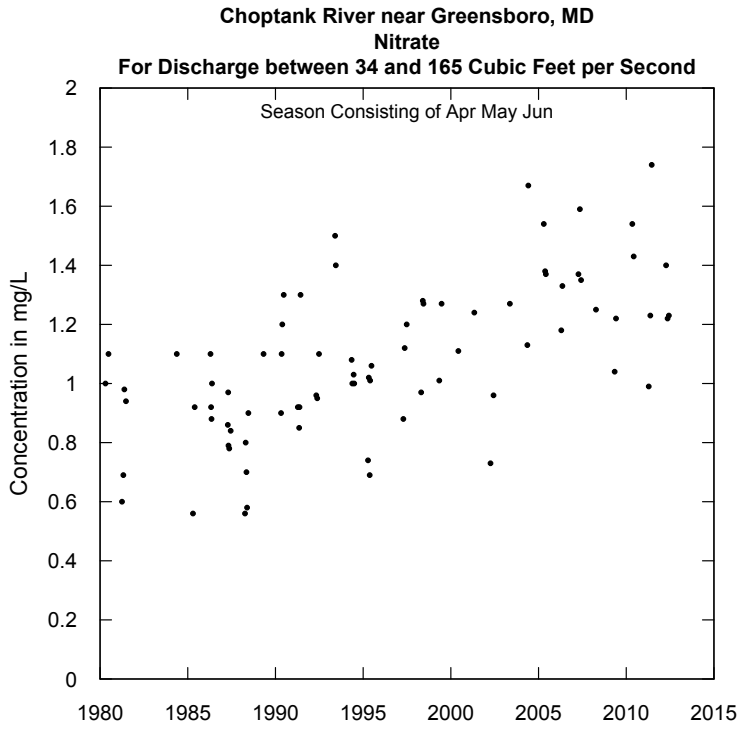


Figure 15. Plot produced by the `plotConcTime` function showing nitrate concentration in milligrams per liter (mg/L) for April, May, June and for discharge between 34 and 165 cubic feet per second for the Choptank River near Greensboro, Maryland.

the command specifies that the plot shows all the concentration data collected on days with discharge in the range of 34 to 165 cubic feet per second (ft^3/s) during the three month period starting in April. The results are in figure 15.

Note that the graphic is entirely self-labeled, and the station name, parameter, season, and discharge range are all included in the figure title. Self-labeling can be very useful when generating many plots, because it ensures that the specific set of conditions used for making the plot are not forgotten later on. In this example, the selected lower and upper bound on discharge are the 25th and 75th percentiles on the flow-duration curve for the full data set. This flow-duration information is determined by the function `flowDuration,` which is described below. The interpretation that can be taken away from this figure is that, in this mid-range of discharge, the trend in concentration over the 33 years has been rather consistently upwards. In contrast to this plot, a different command can be given to produce a graph (fig. 16) of all of the concentrations greater than 165 ft^3/s. The command is:

```
plotConcTime(eList, qUnit=1,qLower=165, concMax=2)
```

Note the addition of the argument `concMax = 2`. This argument is entered to ensure that the maximum value on the y-axis will be 2 mg/L, like the scale used in figure 15. Setting the upper limit on the vertical axis enables the reader to compare levels and (or) trend slopes across multiple graphs in a given report.

What we see in figure 16 is that there is really no appearance of a trend in concentration for this season and discharge range, and that concentrations in this discharge range in the later years of this record are lower than the concentrations that occur at lower discharges (fig. 15). These observations lead to a hypothesis that agricultural practices in this watershed are continually causing increased loadings of nitrate to the groundwater system. This watershed and factors that influence water quality there are described in (Sprague and others, 2000, pages 85–93). That increase is then expressed via the surface-water quality at moderate streamflow, when most of the stream water is derived from the groundwater system. During higher flows, however, the pathways for nitrate are more directly through surface-water flow to the stream, and it appears that improved agricultural practices may be effective in limiting the input of nitrate during these events. A later section of this report (“Exploring model behavior and adjusting model parameters”) contains examples of how these various types of trends (by season or by flow class, or both) can be identified and depicted.

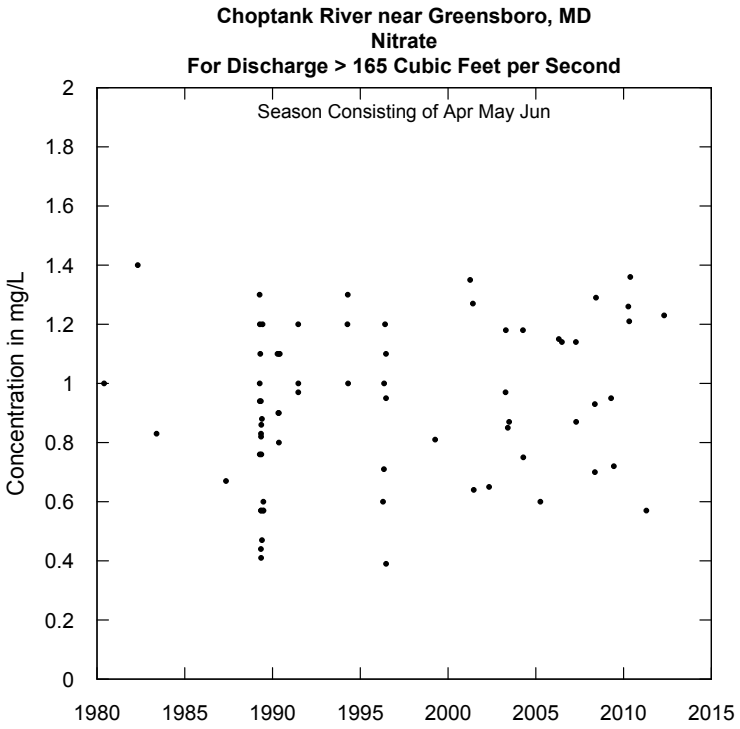


Figure 16. Plot produced by the `plotConcTime` function showing nitrate concentration for April, May, June and for discharge greater than 165 cubic feet per second for the Choptank River near Greensboro, Maryland.

flowDuration

This function is not a graphical function, but it is very useful for setting up graphical functions. It provides the user with context about the range of discharge variables that are common and those that are extreme, for the record as a whole or for a particular part of the year. Figure 17 presents the output from the command:

```
flowDuration(eList, qUnit = 1)
```

```
Flow Duration for Choptank River near Greensboro, MD

Flow duration is based on full year

Discharge units are Cubic Feet per Second
    min      5%     10%     25%     50%     75%     90%     95%     max
   0.35   11.00   16.00   34.00   87.00  165.00  292.00  465.00 8700.00
```

Figure 17. Output from the `flowDuration` function for the Choptank River near Greensboro, Maryland.

The `flowDuration` function is designed not only to describe the flow-duration curve for the entire year, but also to define it for particular parts of the year. For example, we see here that 34 ft^3/s is the 25th percentile on the full annual flow-duration curve. Where does a discharge value of 34 ft^3/s rank for the season graphed in figure 15? This can be determined by using all of the arguments in flowDuration, and the command would be:

```
flowDuration(eList, centerDate = "05-16" , qUnit = 1 , span = 45)
```

The argument `centerDate` specifies the month and day that are at the center of the time span of interest (May 16, being the central date of the April, May, June period), `qUnit` specifies results in ft^3/s, and the `span` argument is half of the width of the period of interest. In this example, the days included in the computation are the `centerDate` plus those days that are less than or equal to 45 days on either side of it (for a total of 91 days). The output shown in figure 18 reflects all the choices made through the selection of these arguments when giving the results.

Thus, we can see that for this portion of the year, a discharge of 34 ft^3/s is slightly above the 5th percentile of the flow-duration curve, rather than the 25th percentile for the full annual distribution.

```
Flow Duration for Choptank River near Greensboro, MD

Flow duration period is centered on May 16
And spans the period from April 1  To June 30

Discharge units are Cubic Feet per Second
   min     5%    10%    25%    50%    75%    90%    95%    max
   8.6   30.0   40.0   65.0  112.0  190.0  336.0  526.0 4170.0
```

Figure 18. Output from the `flowDuration` function, with span equal to 45 days and May 16 as the center date, for the Choptank River near Greensboro, Maryland.

plotConcQ

A fundamental part of any analysis of surface-water quality is to develop an understanding of the relation between discharge and concentration. This relation lies at the center of the WRTDS model as well as other commonly used approaches to water-quality analysis such as LOADEST (Cohn and others, 1992) or the Seasonal Kendall test on flow-adjusted concentrations (Hirsch and others, 1982). A scatterplot of this relation is produced by using the function `plotConcQ`. The graphic shown in figure 19 was created with the command `plotConcQ(eList)`.

The key arguments that are available for modifying this plot (other than those that affect labels, fonts, and colors) are these:

- `qUnit`, which determines the units for plotting discharge values.
- `logScale`, if `TRUE` the vertical axis (concentration) is a log scale, if `FALSE` (the default value) it is an arithmetic scale with the origin at zero.
- `concMax`, which sets the maximum value for the vertical scale and which can be useful when there is reason to produce a set of similarly scaled plots (for example across multiple sites); the default is that the program will set the maximum based on the actual maximum value.
- `concMin`, which sets the minimum value for the vertical scale. It is used only when `logScale=TRUE`. The default, when `logScale = TRUE`, is that it is automatically set based on the data. When `logScale = FALSE`, the default is always zero.

Use of the `logScale` option can be very helpful, because the graph is then presented in the same space in which the WRTDS model will be fitted (log of concentration versus log of discharge), and the common functional forms used in LOADEST or related models will always be linear or quadratic when plotted this way. For example, figure 20 was produced with the command: `plotConcQ(eList, logScale=TRUE)`

Plotting in this manner can often be helpful for identifying a clear characterization of the relation between discharge and concentration; for example, in this case it shows rather clearly that there is little systematic relationship between concentration and discharge for low discharge values, below about 5 m^3/s. Then, beyond that discharge there is a substantial and fairly linear downwards slope. This suggests that base flow water has a rather constant concentration, in the range of 0.5–2 mg/L, but as storm flow begins to enter the system (either as surface runoff or shallow groundwater movement during and just after precipitation), the new water appears to have a lower nitrate concentration than the base flow water, resulting in a dilution effect. It is likely that neither a parabola nor a linear relationship (in log-log space) would be a good representation of this relationship. Just like `plotConcTime`, this graphic can be used to show data from any specified PA by using the `setPA` function. This can be very helpful for exploring seasonal differences in the flow versus concentration relation.

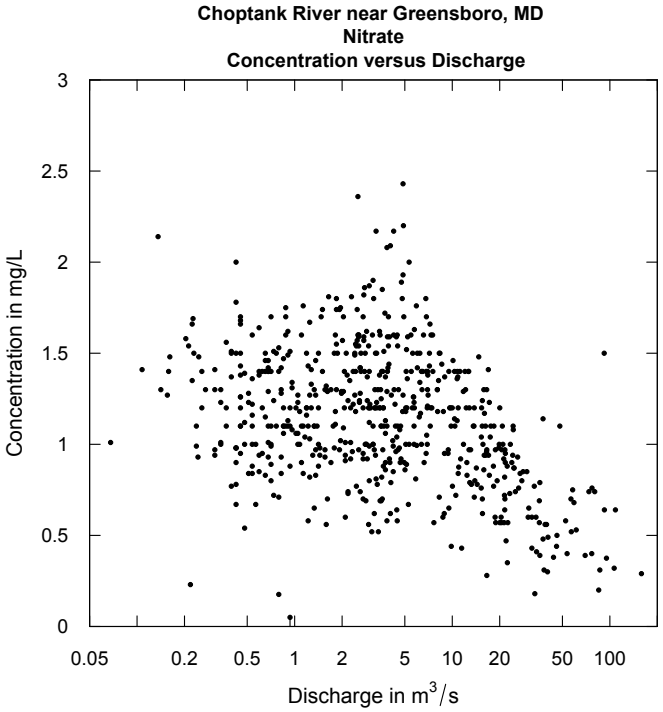


Figure 19. Plot produced by the `plotConcQ` function showing the relation of nitrate concentration and discharge for the Choptank River near Greensboro Maryland.

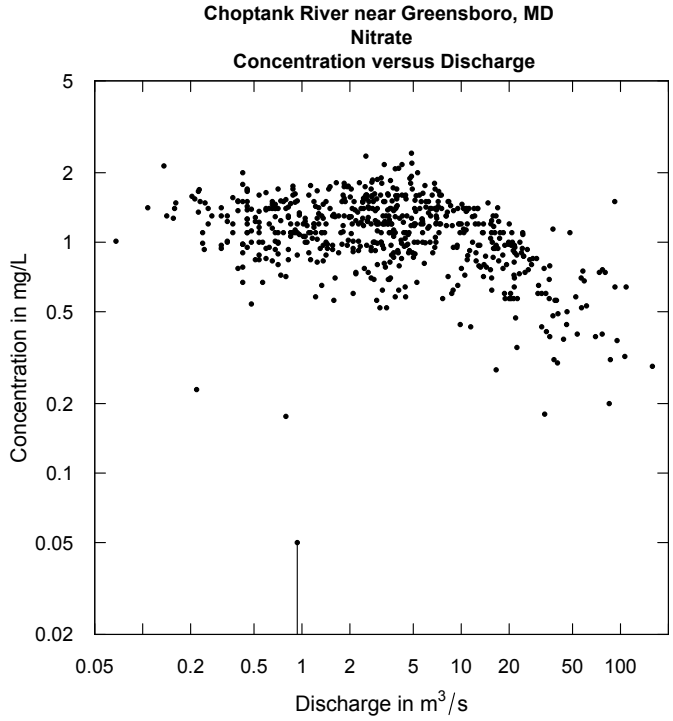


Figure 20. Plot produced by the `plotConcQ(eList, logScale=TRUE)` function showing the relation of nitrate concentration on a log scale and discharge for the Choptank River near Greensboro Maryland.

plotFluxQ

The `plotFluxQ` function produces a graphic that is very closely related to that shown in figure 20. The graphic produced by `plotFluxQ` simply presents the data set as flux values rather than as concentrations. For example, the same data set used for figure 20 can be plotted with the command: `plotFluxQ(eList, fluxUnit=4)`, and the results are shown in figure 21. The choice of units, in this case thousands of kilograms per day (kg/day), is a matter of personal preference.

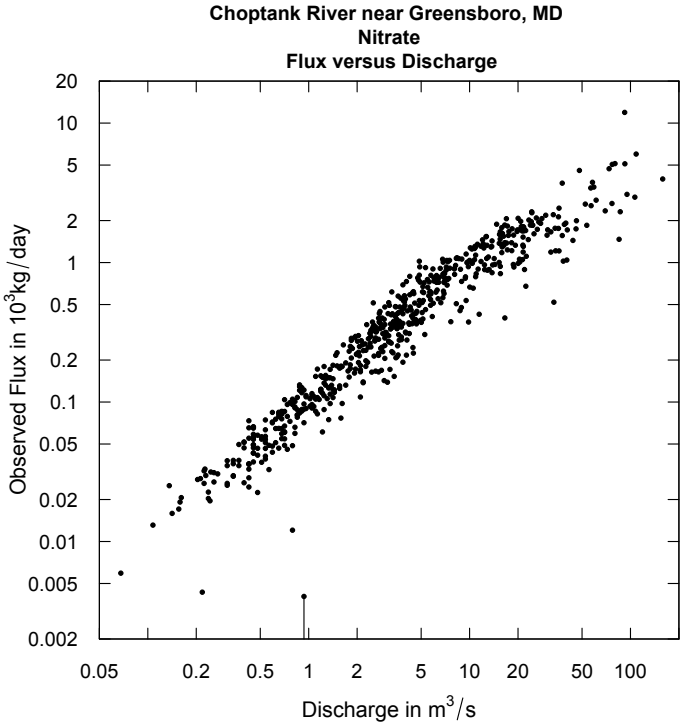


Figure 21. Plot produced by the `plotFluxQ function` showing the relation of flux and discharge for the Choptank River near Greensboro Maryland.

This plot in figure 21 shows the same change in slope that is seen in figure 20. It demonstrates that flux does continue to rise with increasing discharge, but the slope declines with discharges greater than about 5 m^3/s.

boxConcMonth

The `boxConcMonth` graphical function produces boxplots by month and is a simple way to characterize the amount of seasonal variability in the data and learn, at a glance, what times of the year have particularly high or low concentrations. Figure 22 is an example of a boxplot produced for a site that has a very strong and rather complex seasonal pattern (it is based on data from the Iowa River at Wapello, Iowa; the watershed has extensive corn and soybean agriculture). Note that the boxplots used in EGRET all follow the standard conventions for the length of the box and the length of the whiskers as defined in the R-documentation for the function boxplot (enter `?boxplot.stats` in the console during an R session).

The graphic shows a pattern of high nitrate concentrations during the spring period of high runoff, moderately high concentrations throughout the colder months of the year (November–March), and relatively low concentrations in the late summer (August and September). This pattern is a result of low discharge combined with high temperatures, which causes denitrification to remove nitrate from the river very effectively. A variety of studies that discuss the factors that drive these summertime declines are summarized in (Hirsch, 2014). The graphical tools in EGRET are designed to aid in exploration for specific seasonal or flow-related drivers of water quality.

The boxplots presented here use the plotting convention that the width of the box is proportional to the square root of the number of samples represented by the box. This visual cue makes it very clear if there are large differences in the intensity of samples across the different months of the year. In this case, we can see that September, November, January, and February

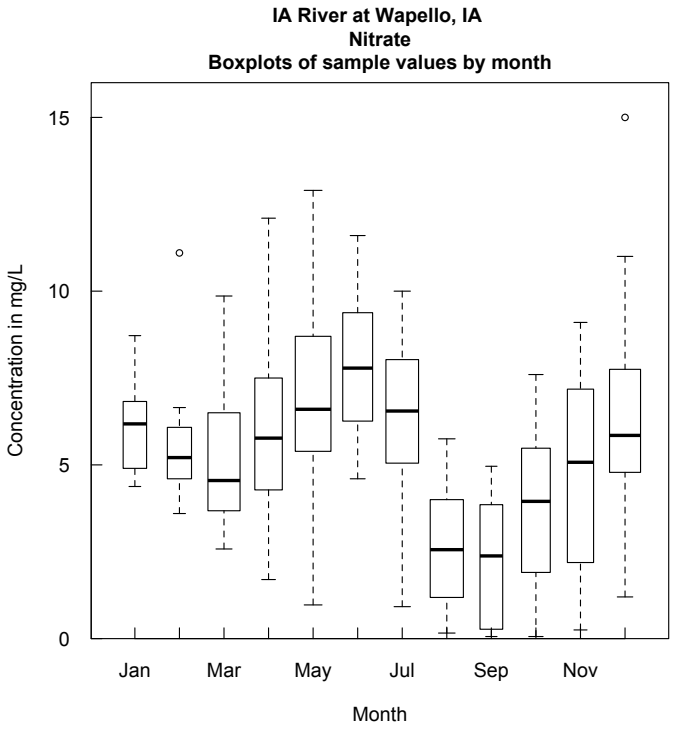


Figure 22. Boxplots produced by the `boxConcMonth` function of nitrate concentration by month for the Iowa River at Wapello, Iowa.

have somewhat fewer samples than the other months. The month with the greatest number is May (37 samples), and the month with the least is September (15 samples). This plot can alert the hydrologist to situations that require considerable care in data analysis because of great differences in sampling intensity; for example, sites with virtually no winter sampling.

boxQTwice

The `boxQTwice` graphical function shows only information about discharge data. It produces two side-by-side boxplots of discharge (on a log scale). The first shows the discharges at which the samples in the data set were collected. The second shows the full population of daily mean discharge values for the entire period of record that will be used in the WRTDS analysis; typically, this should be a period that starts with the first water year of sampling and ends with the last water year of sampling. If the sampling strategy for the water-quality samples were either random or uniform (for example, sample once every 14 days), then the two boxplots should be very similar to each other, certainly similar in terms of median values and upper and lower quartiles (the bounding lines on the box). From the standpoint of being able to make accurate assessments of flux (by WRTDS or other methods), it is desirable that the sampling strategy place more emphasis on collecting samples at high discharge. Figure 23 is an example showing a sampling pattern that is well suited to obtaining accurate flux estimates. The lower quartile, median, and upper quartiles of the distribution of sampled discharges are all offset higher than the discharges in the full record for daily mean discharge. Also, the very highest sampled discharge is actually the third highest daily mean discharge value of the period of record, and the daily mean discharge on the highest day is only 36 percent higher than the highest sampled day. This type of plot can be helpful for identifying those cases where high discharge days are seriously undersampled and, as a consequence, the fluxes estimated for those cases should be treated with a high degree of caution. These boxplots also use the convention of box width being proportional to the square root of the sample size.

multiPlotDataOverview

The graphic produced by `multiPlotDataOverview` is a single multipanel plot that incorporates the results produced by the four specific plotting functions: `plotConcQ, plotConcTime, boxConcMonth, and boxQTwice`. Figure 24 is an example produced by using nitrate data from the Iowa River at Wapello, Iowa. At a quick glance, one can learn several things about the data set:

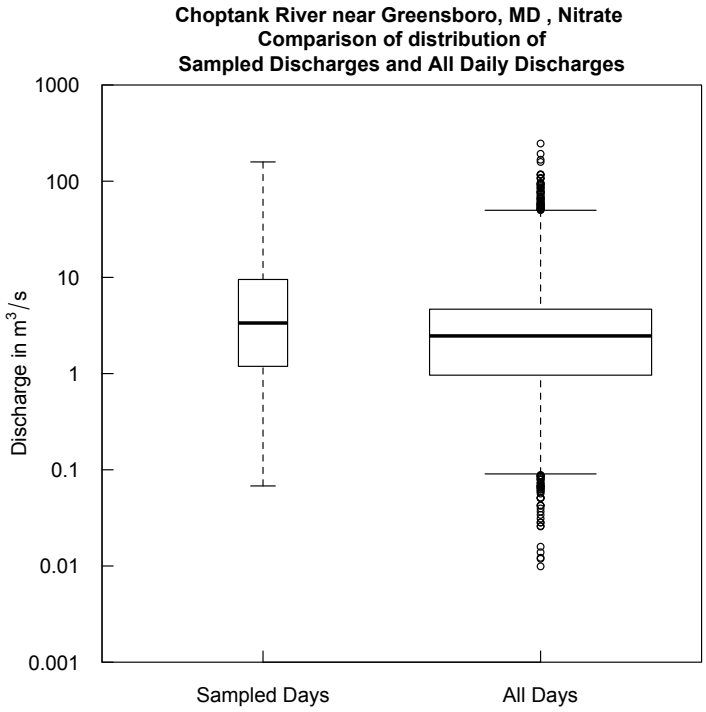


Figure 23. Output from the `boxQTwice` function for the Choptank River near Greensboro, Maryland.

1. concentration increases with discharge—but only up to a point—and then shows some indication of decreases at the highest discharges (dilution with lower concentration precipitation water);
2. the variability of concentration in log space is much greater for low discharges than for high discharges,
3. there is no easily discernible trend;
4. the concentrations are highly seasonal, with very low values in the late summer and very high ones centered around April; and
5. the sampling is fairly well distributed across all months and well distributed across the range of discharges, with some bias towards sampling of higher discharges, which is a desirable situation (as discussed above).

All of the component plots can be used with a PA less than the full year; thus, `multiPlotDataOverview` can be produced for any PA and will be labeled accordingly.

WRTDS Analysis of Water-Quality Data

WRTDS (Weighted Regressions on Time, Discharge, and Season) is a method for analysis of water-quality data sets that can be used to characterize the status and trends in concentration and flux. For an extensive discussion of the motivations and design of the WRTDS method, see Hirsch and others (2010). The method can be used for a variety of purposes including:

1. estimating long-term changes (trends) in average concentrations and average fluxes, both annually and for some selected PA;
2. estimating actual mean concentration or fluxes for specific years or specific PAs within the year;
3. estimating mean concentrations or mean fluxes over some specified period, such as a decade; and
4. for providing insights into the change in system behavior that may lead to a better understanding of the causative mechanism behind the trends that are observed.

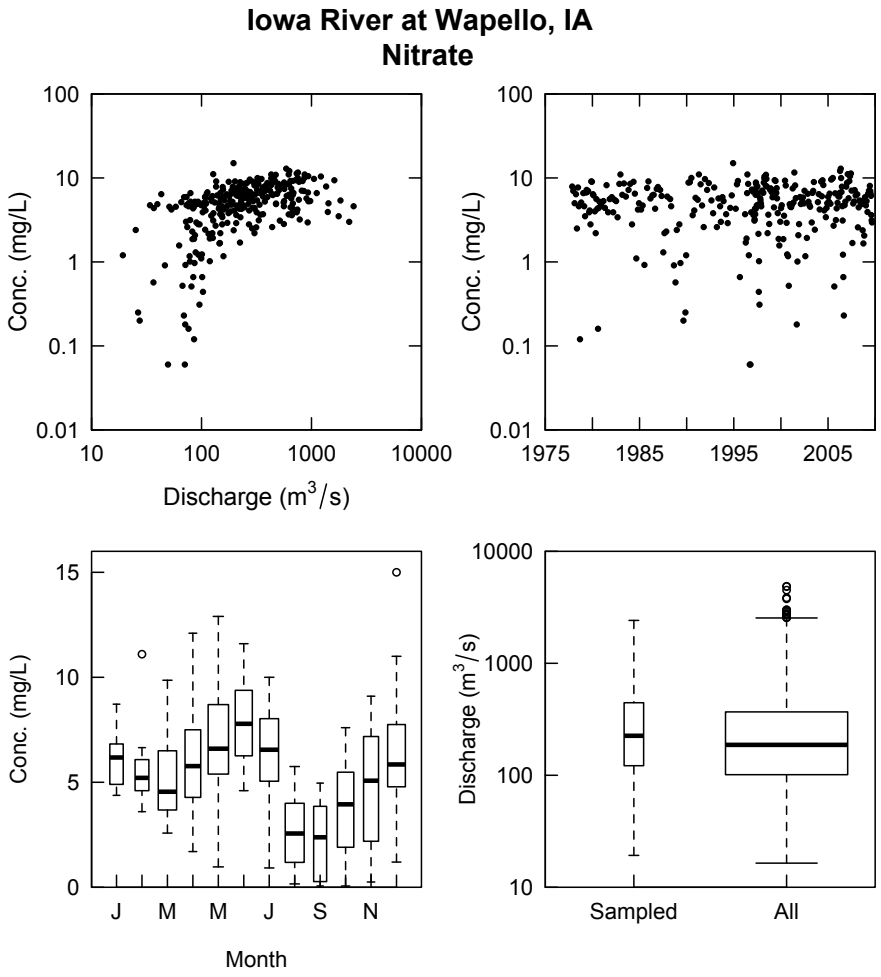


Figure 24. Output from the `multiPlotDataOverview` function for nitrate data from the Iowa River at Wapello, Iowa.

The method requires the availability of measured concentrations of the constituent of interest and a complete record of daily mean discharge for some period of record. The method was designed for data sets with 200 or more measured concentration values that span a period of a decade or more. However, testing has shown that in some cases, it can produce reliable estimates of mean concentrations or mean fluxes with data sets as small as about 60 samples spanning periods as short as a decade, but doing so requires special settings on some of the arguments of the `modelEstimation` function (described below). The method is not appropriate for small flashy watersheds where discharge at the streamgage commonly changes by an order of magnitude or more within a given day, although it can be appropriate for rivers that are subject to regulation at time scales of less than a day. The discharge data set must cover the entire period of water-quality data used, although it should not be more than a few months longer than the water-quality record. Use of a discharge record that extends substantially before and (or) after the period of the water-quality record will result in highly unreliable estimates for the years before and (or) after the water-quality record. Smoothing methods, such as WRTDS, should never be used to make extrapolations. A good rule of thumb is to have the discharge record start at the beginning of the first water year for which there are water-quality samples and end at the end of the last water year for which there are water-quality samples. The data used in the model estimation process are all stored in `eList`, which contains three data frames: `Sample`, `Daily`, and `INFO`. All of the sample data are stored in the `Sample` data frame, the discharge information is stored in the `Daily` data frame, and the metadata are stored in the `INFO` data frame (see the "Data Entry" section of this report).

The WRTDS method creates a highly flexible statistical representation of the expected value of concentration for every day in the period of record and then uses that representation to produce four daily time series for the period of record. These are daily concentration, daily flux, flow-normalized daily concentration, and flow-normalized daily flux. The flow-normalized values are intended to describe the changing state of the system over time, by integrating out the influences of variations in concentration or flux that arise from the day-to-day variations in discharge. In contrast, the nonflow-normalized versions of these variables are

estimates of what actually happened on each day and, as such, they are highly influenced by the actual discharge that happened that day.

Estimates of Concentration and Flux

The WRTDS model can be thought of as a smooth surface that describes

$$E[c] = w(Q,T) \tag{7}$$

where

c is concentration, in mg/L,
$E[c]$ is the expected value of concentration,
w is a function that depends on two variables,
Q is discharge, in m^3/s, and
T is time, in decimal years.

This function can be illustrated in the form of a contour plot, where the horizontal axis is time, the vertical axis is discharge (shown on a log scale), and the contour plot shading is based on the value of $E[c]$. The graphic shown in figure 25 represents the estimates of w over a rectangular grid of Q and T values for chloride concentration in the Milwaukee River at Milwaukee, Wisconsin. The figure indicates that, over this 35-year period, concentrations have been highest at low flows, there is seasonality such that, for a given discharge, concentrations are higher in the winter than in the summer, and there is an overall upwards trend in concentrations across most of the range of discharges and seasons. Figure 25 was produced by the `plotContours` function, which is described in detail in the section "plotContours."

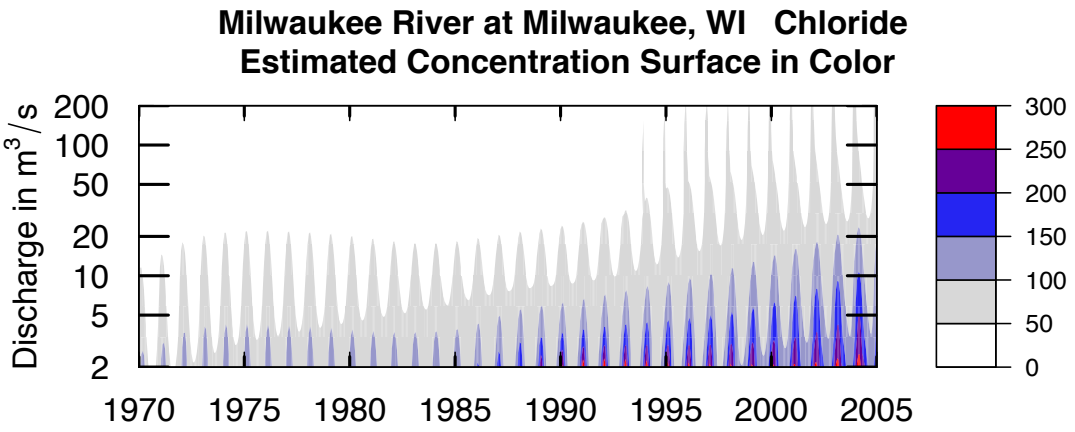


Figure 25. Contour plot of expected value of chloride concentration as a function of time and discharge, Milwaukee River at Milwaukee, Wisconsin.

In the EGRET code, the discharge values for the grid are 14 values that are equidistant in log space. They run from a discharge value about 5 percent below the minimum discharge in the daily record to a value about 5 percent above the maximum discharge in the daily record. This is defined more precisely below in the discussion of the `surfaceIndex` function. The grid values for time are spaced 0.0625 years apart ($1/16^{th}$ of a year). The full data set used in this example actually extends beyond the bounds of the graphic shown, consisting of 673 time steps by 14 discharge steps for a total of 9,422 nodes, of which 4,616 fall within the space covered by this graphic. At each node of the grid, the estimates of $E[c]$ are made as follows. A weighted regression model is estimated. It takes the form:

$$\ln(c) = \beta_0 + \beta_1 q + \beta_2 T + \beta_3 \sin(2\pi T) + \beta_4 \cos(2\pi T) + \varepsilon \tag{8}$$

where

c is concentration, in mg/L,
β are the regression coefficients
q is $\ln(Q)$ where Q is daily mean discharge, in m^3/s,

T is time, in decimal years, and
ε is the error (unexplained variation).

Because equation 8 is fitted at each node, there are unique estimates of each of the β values and a unique value of the standard error at each node. The EGRET program does not save the individual β values, but it does save the estimate of ln(c) at that node and the standard error at that node. It is important to recognize that although the form of the equation is written as if ln(c) is linear in q and T and varies seasonally as a perfect sine wave these properties hold true only locally. Because the coefficients vary throughout the Q,T space, the linearity and sine wave form are free to change substantially over the entire Q,T space. The estimation method assures that the estimates of ln(c) vary smoothly with q and T but are not constrained to linear or sine wave characteristics.

This equation is actually fit in the form of a weighted Tobit model (Tobin, 1958), where the c values can be individual values of concentration, or they can be intervals such as (0,0.05) where 0.05 is the minimum reporting level, (one would typically call such a value of concentration "less than 0.05"). Alternatively, they could be some other interval such as (0.04,0.05), which might arise when the analyte in question is the sum of two individual analytes (see appendix 2, section 3.2.4, for a discussion of data entry for these censored cases).

The weights on each of the individual observations in the data set are determined based on three metrics of distance between the node and the specific observation: distance in time, distance in q, and distance in season, which is measured in units of years, but only considers the fractional part of the time separation in years. For example, if we compare a sample value with q and T values of 3.0 and 1995.0, respectively, to a grid point with q value of 3.8 and a T value of 1997.25, then the distance in log discharge would be 0.8, the distance in time would be 2.25 years, and the distance in season would be 0.25 years. Weights are associated with each of these three distance measures by using the tricube weight function (described above in the section titled "The Smoothing Method Used in Flow History Analysis"). The half window widths (the h value in the formula for the weights) have default values of 2 (in log discharge units), 7 years, and 0.5 years. The half window width for time follows the same `edgeAdjust` convention as is described in the "Flow History Analysis" section, causing the half-window width to become wider for estimation points close to the beginning or end of the record. The overall weight for any observation is the product of the three weights, so a weight of zero for any of the three metrics results in an overall weight of zero. For any given node on the grid, the estimate of ln(c) is computed (by using the R function `survReg`, which is an implementation of "survival" or Tobit regression). In the EGRET code, this estimated value is known as `yHat`. In addition, the scale parameter of the survival regression is also stored for every node. The scale parameter is equivalent to a standard error (SE) of the residuals in an ordinary multiple regression. This error measure is known as `SE` in the EGRET code. To determine the expected value of c, the `yHat` value must be multiplied by a bias correction factor (BCF) to account for the fact that the model is estimating ln(c) rather than estimating concentration itself, and the errors in these estimates of ln(c) are assumed to be normally distributed. The BCF at each grid point is exp(SE^2/2). The third result that is stored is $E[c]$, is called `concHat` in the EGRET code. These three results are related by this formula:

$$concHat = BCF \bullet \exp\left(yHat\right) \tag{9}$$

Figure 26 shows each of these three surfaces (`concHat`, `yHat`, and `SE`, each as a function of `DecYear` and `Q`). The bottom panel in figure 26 shows a 2-year segment of the full history shown in figure 25. The following are several observations about these surfaces.

- All of the surfaces are relatively smooth; this is a consequence of the weighted regression smoothing approach. Modifying the half-window widths would change the degree of smoothing in these surfaces. The section "Exploring model behavior and adjusting model parameters" considers the basis for selecting appropriate half-window widths.

- They all show a seasonal pattern, but that pattern evolves slightly over time. In general, that pattern is one of the highest concentrations in the winter and the lowest concentrations in the summer. The highest concentrations are focused during times of very low discharge in the months of January, February, and March. During all seasons, concentration tends to decrease as discharge increases.

- The variability (shown in the middle panel as the plot of estimated standard error of log(c)) shows clear differences across a range of discharges and seasons, indicating that an assumption of homoscedastic residuals is not appropriate. This point is an important distinction between WRTDS and LOADEST (Cohn and others, 1992; Runkel and others, 2004). In LOADEST, the errors are assumed to be essentially constant across all seasons and discharges, and consequently, the BCF they use is virtually constant across all seasons and discharges. The WRTDS model recognizes and uses these very substantial differences in the SE to compute the estimated concentrations.

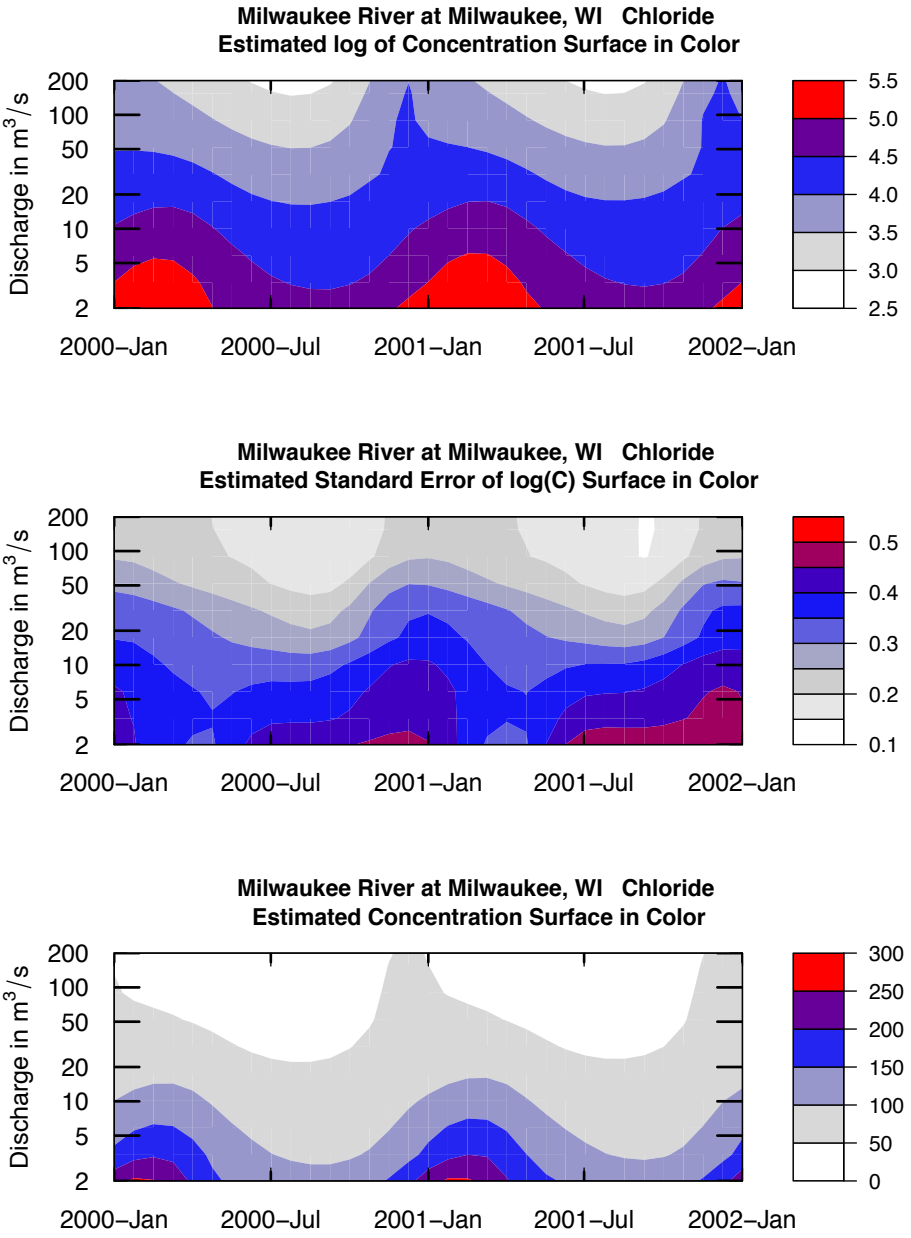


Figure 26. Contour plots of fitted Weighted Regressions on Time, Discharge, and Season (WRTDS) model for chloride data for the Milwaukee River, showing the years 2000–2001. The upper panel is the estimate of the ln(*c*) surface, the middle panel is the estimate of the standard error, and bottom panel is the estimate of concentration, in milligrams per liter.

The WRTDS model uses the characterization of the $E[c]$ surface to make estimates of concentration for every day in the period of record. These individual daily estimates of concentration are made through a bi-linear interpolation of the `concHat` value from the grid of estimates, using the values of q and T specific to that day. If the model assumptions of WRTDS were perfect, then these estimates would each be an unbiased estimate of concentration for that specific day. However, these estimates, taken in aggregate, will not exhibit as much variability as a real record would. As with all regression-based estimates, they have the property that they "regress to the mean."

Each of these daily estimates of concentration is also used to compute a daily estimate of flux. Flux for each day, in units of kg/day, is computed as

$$concHat \bullet Q \bullet 86.4 \tag{10}$$

where

`concHat` is the daily estimate of concentration, in mg/L,

Q is the daily mean discharge, in m^3/s, and

86.4 is the unit conversion.

Figure 27 shows a 3-year example of the concentration and flux outputs of the model and compares the model results to the actual measurement. In EGRET, these daily estimates of concentration and flux are both stored in the `Daily` data frame and are named `Daily$ConcDay` and `Daily$FluxDay`, respectively.

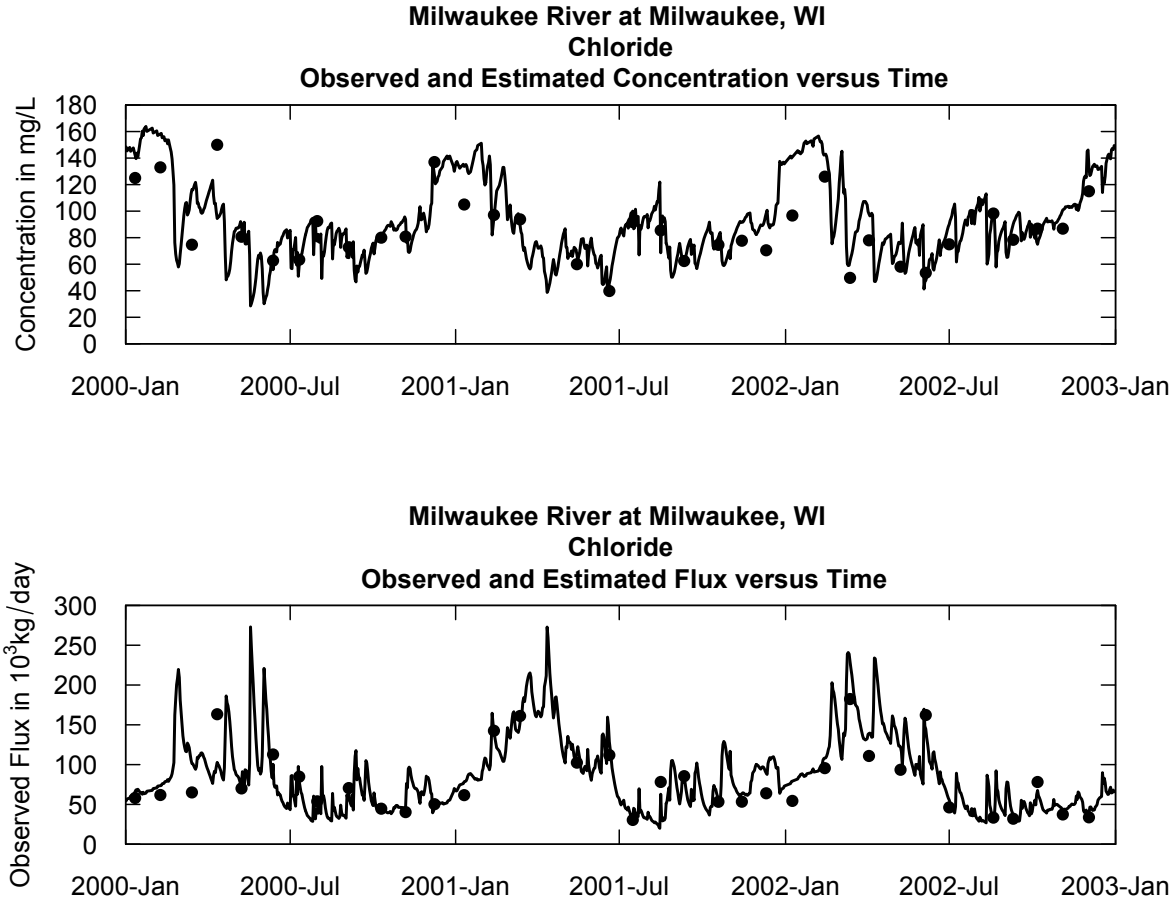


Figure 27. Observed and estimated chloride concentrations (upper panel) fluxes (lower panel) for the Milwaukee River. In each panel, the dots represent the observations and the line represents the Weighted Regressions on Time, Discharge, and Season (WRTDS) estimates for all of the days.

Estimating Flow-normalized Concentration and Flux

Estimates of daily concentration and daily flux are of great value and importance in terms of knowing the actual history of water quality in a river. Particularly where there is an interest in understanding the water quality or ecological condition in a receiving water body such as an estuary, lake, or reservoir, the variable of interest would be the history of flux integrated over time periods such as months, seasons, or years. In addition, when the interest is in the concentrations of a pollutant that may have impacts on receptors such as biota or water supply intakes, then the history of concentration will be of interest. However, the history produced by the model will not describe the frequency of exceedances of water-quality criteria or standards, because the concentration estimates will have less variability than real records would. The concentration or flux

estimates (`Daily$Conc` and `Daily$Flux`) can be very strongly influenced by the particular time history of flow conditions. For example, for a pollutant for which concentration increases with discharge, a high-flow period of a year or two near the end of the period of record can suggest deteriorating water quality. For those interested in evaluating the effectiveness of pollution control efforts, these types of results can seriously confound the analysis. The variability in concentration or flux that is related to discharge can overwhelm a true signal of change. Discharge-driven variability creates a large amount of apparent "noise," thus making the identification of trend virtually impossible. For those seeking information about "progress" or "effectiveness" or an understanding of how the watershed system is changing, what is needed are time histories that filter out the impact of year-to-year variation in discharge.

The method used to accomplish this in WRTDS is termed "flow normalization." It can be described in the following manner:

$$E\left[C_{fn}(T)\right] = \int_0^\infty w(Q,T) \bullet f_{Ts}(Q)\, dQ \tag{11}$$

where

$E\left[C_{fn}(T)\right]$ is the flow-normalized concentration for time T (a specific day of a specific year)

$w(Q,T)$ is the WRTDS estimate of concentration as a function of Q (discharge) and T (time, in years)

$f_{Ts}(Q)$ is the probability density function (pdf) of discharge, specific to a particular time of year, designated at Ts.

T_s is restricted to values between 0 and 1, and it is defined as the fractional part of the time variable T (thus T_s is the decimal portion of `decYear`).

Thus, the flow-normalized concentration on a specific day (a specific value of T) is the integral of the fitted estimates of concentration as a function of discharge and time multiplied by the pdf of discharge for that day of the year.

The challenge to operationalizing this function is the specification of the pdf of discharge for each day of the year. In WRTDS, this starts with the assumption that for any given day of the year the distribution of discharge is stationary. This means that, for example, the pdf of discharge for September 9, 2013, is the same as the pdf of discharge for September 9, 1993. If there is a strong reason to believe that discharges have not been substantially stationary over the period of record, then the flow-normalization method is not appropriate. From a practical standpoint, this suggests that flow normalization should not be used if major changes in the watershed took place during the period of record that have the potential for causing a substantial shift in the probability distribution of daily discharges for part or all of the year. Examples of such changes could include a large dam or diversion put into operation or taken out of operation, a large new import of water, a large change in consumptive use of water, or large changes in base flow due to changes in land drainage practices or groundwater levels. The presence of dams upstream is not a reason to avoid using this method unless there was a substantial shift in the operations of the dam that changed the pattern of daily discharges (not subdaily discharges) at the streamgage of interest. In addition, dams built or removed upstream of the streamgage that control a small fraction of the watershed's flow are not reasons to avoid using flow normalization. At present, the changes in discharge due to climate change are not likely to be of sufficient magnitude to invalidate the flow-normalization method, but they may become a more important consideration in future years.

A good test of the appropriateness of the method could come from the hydrologist performing the following thought experiment. On September 9, 2013 (for example), what do you think the pdf of discharge is for September 9, 2014? If you say that the history of discharge on all of the past September 9ths would provide a good estimate, then you should be willing to use this procedure. However, if you believe that the pdf for September 9, 2014, differs from the history in some specific and quantifiable way, then the flow-normalization method should not be used. There are ways to modify the WRTDS method to accommodate this nonstationary case, but the necessary methods have not been developed or tested. Making such enhancements is an important goal for the future of WRTDS and the EGRET software package.

The other issue that must be resolved to use this method is how to characterize the pdf of discharge for each specific day of the year. There are certainly ways that a continuous, seasonally varying model of streamflow distributions can be characterized, but they require a large number of assumptions and estimated parameters. Given the richness of the discharge data sets that are used in WRTDS (a 20-year record has over 7300 values), it is reasonable to simply use the full historical sample as a representation of the pdf. In the case of a 20-year record, this means that we would assume that the pdf for September 9 is the 20 observed values, and each is assigned a probability of 0.05. The resulting pdf by itself is not very realistic (having zero probability for all values other than these), but when taken together with a similarly constructed pdf for September 10, September 11, etc., the collection of these daily pdfs begins to take on the characteristics of a nearly continuous and smooth distribution of discharge. The flow-normalized estimates for successive days can differ from each other by large amounts, but WRTDS flow-normalized estimates are aggregated to averages for a specific months, seasons, or years. The approach described here produces time series of annual flow-normalized values that are quite smooth because they integrate over such a large number of individual daily estimates.

In practical terms, how is this computation carried out? If our interest is in the flow-normalized concentration for September 9, 2013, then we use the $w(Q,T)$ function evaluated at T of 2013.690 (because September 9 is 0.690 years from the start of the year), so this becomes a function simply of Q. Then, we assemble the Q values for all 20 of the September 9ths in the record and evaluate $w(Q,2013.690)$ by using bilinear interpolation in terms of T and $\ln(Q)$. The average of these 20 concentration values becomes the flow-normalized concentration value for September 9, 2013, which is the expected value of concentration on September 9, 2013, integrating over the estimated frequency distribution of discharge for September 9. This process is repeated for every day in the period of record, and the results of the process become the time series of flow-normalized concentrations for the period of record. These daily values can be aggregated to monthly values by computing a mean of the flow-normalized values for the month and to yearly values by computing a mean of the flow-normalized values for the year. These monthly values will show strong seasonality and will change gradually over time, but they will be free of any variation due to the occurrence of high- or low-flow conditions in any given month. Similarly, the annual values will show gradual change over time, but will be free of any variation due to the occurrence of high- or low-flow conditions in any given year.

The flow-normalized flux is computed in a similar manner, but in this case, the random variable of interest is flux rather than concentration. As such, the integral is this:

$$E\left[F_{fn}(T)\right] = \int_0^\infty Q \bullet 86.4 \bullet w(Q,T) \bullet f_{Ts}(Q)\, dQ \tag{12}$$

The EGRET code computes the flow-normalized concentration and flow-normalized flux for every day of the record and stores these in `Daily$FNConc` and `Daily$FNFlux`, respectively.

Fitting the WRTDS Model

The fitting of the WRTDS model by using the EGRET code can be accomplished through a single function, `modelEstimation`. This function takes information from the `INFO`, `Daily`, and `Sample` data frames, which are all stored in the list `eList`, then augments each of them with additional information, creates an additional object called `surfaces`, and returns a new version of `eList` that contains the new versions of `INFO`, `Daily`, and `Sample` plus the newly created `surfaces`. Even though the user can accomplish this through a single command, this section of the report will briefly describe each of the functions that are called by it. The command for conducting the model estimation process is this:

```
    eList <- modelEstimation(eList, windowY = 7, windowQ = 2, windowS = 0.5, minNumObs
= 100, minNumUncen =50, edgeAdjust = TRUE)
```

To invoke the function while leaving all six arguments set to their default values, the command would simply be `eList <- modelEstimation(eList)`. Table 7 explains the six arguments that are used in fitting the model.

Table 7. Information about the seven arguments used in `modelEstimation`.

Argument	Definition and discussion	Default value
`eList`	Specifies the named list that contains the `INFO`, `Daily`, and `Sample` data frames that will be used by this function.	No default, name of list must be stated.
`windowY`	The half window width for the time weighting, measured in years. Values much shorter than 7 usually result in many oscillations in the system that are likely not very realistic. Values greater than 7 may "oversmooth" the underlying trends.	7
`windowQ`	The half window width for the weighting in terms of $\ln(Q)$. For very large rivers with average discharge values in the range of many thousands of m^3/s, a value less than 2 may be appropriate, but a value less than 1 probably would not be appropriate.	2
`windowS`	The half window width for the seasonal weighting, measured in years. When `windowS` = 0.5, all data points get a nonzero weight, but those close to a half a year away get weights that are virtually zero. For large data sets, a value less than 0.5 may be appropriate, but the value should not be less than 0.25. Values greater than 0.5 cause the importance of the seasonal weighting to be diminished. Setting `windowS` to a large number such as 10 has the effect of eliminating seasonal weighting.	0.5

Table 7. Information about the seven arguments used in `modelEstimation`.—Continued

Argument	Definition and discussion	Default value
`minNumObs`	This is the minimum number of observations with nonzero weight that the individual weighted regressions will require before they will be used. If there too few observations, the program will iterate, making the windows wider until the number increases above this minimum. The only reason to decrease his value is in cases where the data set is rather small. It should always be set to a number at least slightly smaller than the sample size. Any value less than about 50 is probably in the "dangerous" range, in terms of the reliability of the regression.	100
`minNumUncen`	This is the minimum number of uncensored observations with nonzero weight that the individual weighted regressions will require before they will be used. If there are too few uncensored observations, the program will iterate, making the windows wider until the number increases above this minimum. The only reason to decrease this value is in cases where the number of uncensored values is rather small. The method has never been tested in situations where there are very few uncensored values.	50
`edgeAdjust`	This is a logical variable. If `TRUE` the half window width for time weighting is increased when the estimation time is less than `windowY` years from the beginning or end of the estimation period. It is adjusted so that the full width of the window is equal to 2*`windowY`. If FALSE, the window width is not adjusted near the beginning or end of the data set. If FALSE, the optimal `windowY` value may be closer to 10 rather than 7 years.	TRUE

The `edgeAdjust` option was added to the EGRET software in 2014. It was added because experience across a number of applications of the WRTDS method that were conducted between 2010 and 2014 sometimes showed rather strong curvature in flow-normalized concentration or flux trends in the last few (or first few) years of the study period. In some cases, the addition of one or two more years of additional data showed a reversal of the trend pattern exhibited in the earlier analysis. The `edgeAdjust` option has been shown to limit the severity of such reversals. Using it results in a depiction of trend slopes that are more nearly linear in the early and late parts of the record and avoids abrupt changes in slope. A negative ramification of the `edgeAdjust` option is that WRTDS may be slightly slower to indicate reversals that turn out to be real and which persist as more and more data are added to the record. The decision to use it or not use it is a judgment call on the part of the user and represents a trade-off between stability and sensitivity. Past users of EGRET (those who used Beta test version 1.2.5 or lower) can continue to have it operate in exactly the same manner that it did before the introduction of `edgeAdjust` by setting `edgeAdjust` to `FALSE` and setting `windowY` to the value used in their application. The previous default value for `windowY` was 10 years.

The `modelEstimation` function carries out a series of four operations in the following order:

- `estCrossVal`. This function applies the weighted regressions for all the observations in the data set for purposes of model evaluation. It uses "leave-one-out cross validation" to compute, for each of the observations in the `Sample` data frame, 1) an estimate of the log of concentration, 2) the standard error of the regression, and 3) the unbiased estimate of concentration. For each observation in the data set, it runs the weighted regression estimate for that specific discharge and time, but with that specific observation left out of the data set. This provides a more realistic evaluation of the model's ability to make predictions. This "leave-one-out cross validation" is appropriate for a method such as WRTDS, because the highly flexible nature of the method can be overfitted to the data, particularly to the more extreme values in the data set. For a discussion of this approach to cross-validation see, for example, the discussion of the PRESS statistic in Montgomery and others, (2012, p. 151–152). These estimates are added to the Sample data frame and are named: `Sample$yHat` (the estimate of ln(c)), `Sample$SE` (the estimate of the standard error of the regression), and `Sample$ConcHat` (the estimate of concentration). These results are used in a number of functions that are introduced later in the section "Exploring the Quality of the Fitted Model."

- `surfaceIndex`. The purpose of this function is to set up the grid over which the WRTDS model will be estimated. The six parameters that define the grid are determined as a function of the information in the `Daily` data frame: specifically the maximum and minimum daily mean discharge values (`Daily$Q`) and the starting and ending dates of the daily values record (the first and last values of `Daily$DecYear`). These four variables are used to establish the grid used to define the three surfaces that will be stored in the object called `surfaces` (described below). The grid for discharge is equally divided in ln(Q) and runs from the minimum value of ln(Q) minus 0.05 to the maximum value of ln(Q) plus 0.05. This results in a range from 4.877 percent below to 5.127 percent above the range of observed dis-

charges in the full discharge record. The grid for time runs from the start of the calendar year that includes the minimum value of `Daily$DecYear` to the end of the calendar year that includes the maximum value of `Daily$DecYear`. For example, if the discharge record stored in the `Daily` data frame ran from 1982-10-01 to 2012-09-30, then the grid would run from 1982.0 to 2013.0. The grid spacing is 0.0625 years (1/16th of a year), so in this case, the time values for the grid would total 497 values (16 per year times 13 years plus one additional grid for the end of the last year). In this case, the total number of nodes to the grid for computing the surface would be 6,958 (497 columns times 14 rows). The parameters that define the grid are then stored in the `INFO` data frame. They are named: `bottomLogQ, stepLogQ, nVectorLogQ, bottomYear, stepYear,` and `nVectorYear` (signifying the lowest value, the step between values and the number of values, first for the discharge dimension and second for the time dimension). Also stored in `INFO` at the same time are the five specified parameters: `windowY, windowQ, windowS, minNumObs, minNumUncen and edgeAdjust`. By storing all of these parameters in `INFO`, they serve to document how the set of estimates was made.

- `estSurfaces` is the function that fits the model at every one of the grid points defined in `surfaceIndex`. At every grid point, it stores three values into a matrix called `surfaces`. The dimensions of `surfaces` are [`nVectorLogQ`, `nVectorYear`, 3], thus the dimensions in the previous example would be [14, 497, 3]. The first two indices simply go from the smallest to the largest grid values of log(*Q*) and time, respectively. The third dimension stores results in this order: first is the estimated log concentration (`yHat`), second is the estimated standard error (`SE`), and third is the estimated concentration (`ConcHat`). These three values, at any grid node, are related to each other in the following way: `ConcHat` = BCF * exp(`yHat`), where BCF = exp($SE^2/2$). The term BCF is the "bias correction factor." The matrix `surfaces` is used later to create the estimated daily values of concentration, flux, and the flow-normalized versions of these values, and to produce the contour surfaces such as those seen in figures 25 and 26. The `estSurfaces` function calls the function `runSurvReg`, which sets up the survival regression where the estimation is accomplished. The fitting computations are done through the R function `survreg`, which is in the `survival` package that uses `interval2` censoring, which allows for left censoring and interval censoring), with a `lognormal` distribution assumed for the y variable, which is concentration.

- `estDailyFromSurfaces` is the function that uses the `surfaces` object with the individual daily discharge values stored in the `Daily` data frame. For the given values of `DecYear` and `LogQ`, it interpolates values of `ConcHat`. The value of `ConcHat` for a given day becomes `Daily$ConcDay (in mg/L)`, which is then multiplied by the discharge and the unit conversion (86.4) to become `Daily$FluxDay (in kg/d)`. The `estDailyFromSurfaces` function also uses the same interpolation method to compute daily values of `yHat` (the expected value of ln(*c*)) and `SE` (the regression standard error). These are stored as `Daily$yHat` and `Daily$SE`. This function also computes the flow-normalized values of concentration and flux for the day and stores them as `Daily$FNConc` and `Daily$FNFlux`. The process for computing the flow-normalized series is described above in the section "Estimation of Flow-normalized Concentration and Flux."

Displaying and Managing WRTDS Model Results

Computing Annual Results

In the various applications that graph the annual results, provide tables of them, or describe the trends, this step is handled internally by the individual functions. Thus, users generally do not have to take the specific step of computing annual results. However, in some instances users may want to do this if they want easy access to annual time series values in the form of a data frame. The user is free to use any name for this data frame. `setupYears` is the function that takes the contents of `Daily` and computes annual mean values for `Conc`, `Flux`, `FNConc`, and `FNFlux`. These annual average values are then placed in a new data frame called `AnnualResults`. The full set of variables in AnnualResults is shown below in table 8.

Table 8. Column names for the data frame `AnnualResults`.

Name of column	What it contains
`DecYear`	Mean value of `Daily$DecYear` for all the days included in the year.
`Q`	Mean value of `Daily$Q` for all the days in the year, in m^3/s.
`Conc`	Mean value of `Daily$Conc` for all the days in the year, mg/L.
`Flux`	Mean value of `Daily$Flux` for all the days in the year, in kg/d.

Table 8. Column names for the data frame `AnnualResults`.—Continued

Name of column	What it contains
`FNConc`	Mean value of `Daily$FNConc` for all the days in the year, mg/L.
`FNFlux`	Mean value of `Daily$FNFlux` for all the days in the year, kg/d.
`PeriodLong`	The value of `paLong` used to set up the period of analysis (the length of the PA, in months).
`PeriodStart`	The value of `paStart` used to set up the period of analysis (the first month of the PA).

The `setupYears` function allows the user to establish the PA that will be used for computing these annual values. It can be used to compute calendar year or water year results, but it can also be used to compute results for any PA the user chooses. If, for example, the user wanted to consider averages for the months April, May, and June, of each year, the command would be:

```
ApMayJuneResults <- setupYears(eList$Daily, paLong = 3, paStart = 4)
```

Alternatively, if the interest were in just the month of May it would be

```
MayResults <- setupYears(eList$Daily, paLong = 1, paStart = 5)
```

There is no need to reestimate the model (a somewhat time-consuming step) if the user simply wants to change the PA that they use to summarize the results. The user generally does not need to run the `setupYears` function, because the individual functions that present the results as graphs or tables (defined the subsequent section "Displaying and Managing WRTDS Model Results") each make this computation by using whatever PA the user selects.

Given the amount of computer time and effort involved in reaching this point in the analysis, it is wise to save the workspace at this stage. The instructions for doing this are given above in the section "Saving the Workspace for Future Use." The command is simply

```
saveResults(savePath, eList)
```

Computing Monthly Results

If monthly results, rather than annual or seasonal, are desired, they can be generated by using the command: `MonthlyResults<-calculateMonthlyResults(eList)`. The data frame `MonthlyResults` contains mean values for each month for discharge, concentration, flux, flow-normalized concentration, and flow-normalized flux. It also has columns for month (values of 1 through 12 with 1 being January), year (the calendar year), and DecYear (the decimal year value for the midpoint of the month). There are no functions particularly designed to display the content of `MonthlyResults`, but they can certainly be plotted or printed by using standard R functions for plotting or printing the content of a data frame. The units for the results in `MonthlyResults` are m^3/s for discharge, mg/L for concentration, and kg/day for flux.

Issues of Large Data Gaps—Using the `blankTime` Function

When the concentration record has a large data gap, the WRTDS estimates for the time of the data gap are likely to be highly unreliable. Because WRTDS makes no prior assumptions about the shape of the time trend, the computations can create large oscillations during long data gaps. These are just numerical artifacts. A data gap of two years or less (regardless of the overall record length) is generally not a problem, but as gaps become longer, it may be prudent to use the `blankTime` function to eliminate the results for the gap period. The `blankTime` function should also be used if there is a period of a few years during which the sampling frequency is very low; for example, fewer than six observations per year. If there is a long data gap or period of very sparse data, the `modelEstimation` step should be run as usual, followed by running of the `blankTime` function. The `blankTime` function replaces all of the estimated values (`yHat, SE, ConcDay, FluxDay, FNConc, FNFlux`) in the `Daily` data frame during the blank period with `NA`, the indicator for missing values. The user must specify the starting and ending dates of the gap. It may be prudent to make the blank period a few months longer than the actual gap and to start and end it with the starting and ending dates of water years, if water years will be the basis for annual summary computations. Regardless of how the user sets the blank period, any water-quality data that may exist during that period will be used in the estimation process to inform the model, but the model is not used to produce daily estimates during this blank period because they are likely to be highly unreliable. The discharge data during this period are still used, along with the rest of the discharge

data, for making flow-normalized estimates. It is also possible to use `blankTime` more than once on a given data set if it contains multiple data gaps. The command is:

```
eList <- blankTime(eList, startBlank, endBlank)
```

The arguments `startBlank` and `endBlank` must be a date expressed in yyyy-mm-dd form and must be inside of quotation marks. For example, if the user's judgment is that results for water years 1980 through 1989 should be blanked out, then the blank period might be defined as 1979-10-01 to 1989-09-30. In this example the commands would be:

```
startBlank<-"1979-10-01"
endBlank<-"1989-09-30"
eList<-blankTime(eList, startBlank,endBlank)
```

Alternatively, it could be done as a single command.

```
eList<-blankTime(eList, "1979-10-01", "1989-09-30")
```

The result of the command is that a new version of `Daily` that includes `NA` values is substituted for the old version. Any subsequent commands to display annual results as graphs or tables will be based on this new version of `Daily`, and the periods with `NA` values will be reported as missing in the graphical or tabular output.

Plotting Annual Results

The function `plotConcHist` is used to plot the annual average concentration and annual flow-normalized concentration. The annual average concentration is displayed as individual points (plotted at the midpoint of the PA). The flow-normalized results are presented as a smooth curve (in green) even though they are computed for a single point in time for each year. This graphical convention simply emphasizes the relatively smooth nature of the flow-normalized concentration record in contrast with the annual average concentration record.

The `plotConcHist` function can be used to plot the annual average concentration and annual flow-normalized concentration for any PA, and the title always indicates the PA that was used. The user determines the PA by using the function `setPA` prior to running `plotConcHist`. If the `setPA` command has not been used previously for the given data set, then the function produces values for the water year. The command for selecting a different PA would be:

```
eList <- setPA(eList, paStart, paLong)
```

So, for example, to consider the months of April, May, and June, the command would be:

```
eList <- setPA(eList, paStart = 4, paLong = 3)
```

This information about the PA is carried within the `INFO` data frame and is thus used by the other functions described in this section (`plotFluxHist`, `tableResults`, and `tableChange`), but the user can change the PA information at any point during the session. These instructions about the setting up the PA apply in exactly the same manner to the `plotFluxHist`, `tableResults`, and `tableChange` functions.

After the user sets the PA, the command for making the plot, in its simplest form, is `plotConcHist(eList)`. With the arguments that would typically be used to further refine the graphic, the command is:

```
plotConcHist(eList, yearStart, yearEnd, concMax, printTitle, plotFlowNorm)
```

Table 9 describes these arguments.

Table 9. Commonly used arguments for `plotConcHist`.

Argument	Meaning	Default
`yearStart`	The starting year for the graph, typically expressed as integer values; for example, 1980. Note that the left margin of the graph will probably be an earlier year, but the data will commence with the first yearly value after `yearStart`.	NA, which causes the graph to start at or a few months before the first year of record.
`yearEnd`	The ending year for the graph. This should typically be the start of the calendar year after the last annual value to be plotted. The right edge of the graph will typically be after `yearEnd`	NA, which causes the graph to end at or just after the last year of record.
`concMax`	If specified, concMax defines the upper bound on the vertical axis of the graph. This can be very useful if there are going to be multiple graphs of the same constituent for different sites. The use of the same vertical scale facilitates comparisons across sites.	NA, which causes the graph to be self-scaling.
`printTitle`	If TRUE, the title is printed above the graph. If the illustration were for a publication, then this information would go into the caption, and then FALSE would be the best choice.	TRUE, title is printed.
`plotFlowNorm`	If TRUE, the graph shows the annual concentrations as circles and the flow-normalized values as a green curve. If FALSE, it shows only the annual concentrations.	TRUE, both are shown.

The function `plotFluxHist` is used to plot the annual average flux and annual flow-normalized flux. The command, in its simplest form is just `plotFluxHist(eList)`. With the arguments that would typically be used to further refine the graphic, the command is:

```
plotFluxHist(eList, yearStart, yearEnd, fluxUnit, fluxMax, printTitle,
plotFlowNorm)
```

These arguments are identical to the arguments listed above for `plotConcHist`, except that `fluxUnit` is added and the argument `fluxMax` replaces `concMax`. The argument `fluxUnit` defines the units to be used for the vertical axis, and the user can give the command `fluxUnitCheatSheet()` to list all of the options. The list is also provided in appendix 2, section 3.4. The default is `fluxUnit=9`, which is millions of kg/year. If those are the desired units, then the argument `fluxUnit` can be left out. If, for example, the user wants to use thousands of tons per year, then the command should contain `fluxUnit=6`. Flux units should be those that are customary for the audience and be selected so that they do not require too many digits to be printed on the y-axis scale. The argument `fluxMax` is the upper bound for the y-axis of the graph. If the argument is left out, then the graph is self-scaled. The value for `fluxMax` is expressed in the units specified by `fluxUnit`.

Both of these graphics have titles that indicate: the name of the site, the water-quality variable being plotted, the PA being used (for example, water year, or season defined by the months of April, May, and June), and designation of what is being plotted. If `blankTime` has been used, then no points are plotted for the missing years, and the flow-normalized curve has a break during the period of missing data. Examples of these two types of plots are seen in figures 28 and 29. The commands that produced these figures are `plotConcHist(eList)` and `plotFluxHist(eList, fluxUnit=8)`, respectively.

For graphics in a report on multiple sites or for multiple constituents, it may be desirable for the plots to all start and end at the same date, and this can be done by setting `yearStart` and `yearEnd` to the same value in all cases. Note, however, that if this is done, the water-quality and discharge data from before and (or) after the period specified for the graphic have an effect on the results even though these periods are not shown in the graph. Thus, if the goal is to use the same period of record at all sites or for all constituents, then the records used should be edited down to the desired period before running `modelEstimation`. The process for limiting the length of the data set is described below in the section on ”Editing Data Sets.” If the user wants to show the results for the entire period of record for the site and constituent of interest, then the two arguments (`yearStart` and `yearEnd`) can be left out of the command and the limits of the x-axis are set automatically by the data.

Producing Tables of Results

Users may want to publish a printed table of some or all of the results at an annual time step. The function that produces such a table is `tableResults`. It can be used to produce a table that is simply printed to the console, from which the user can copy it and place it into a document, or it can return a data frame, which has all the necessary information to produce a table suitable for entry into a spreadsheet or a word processing document for publication. The process of producing a table in Excel is described in appendix 2, section 12, “Creating tables in Microsoft® software from an R data frame.” The output of

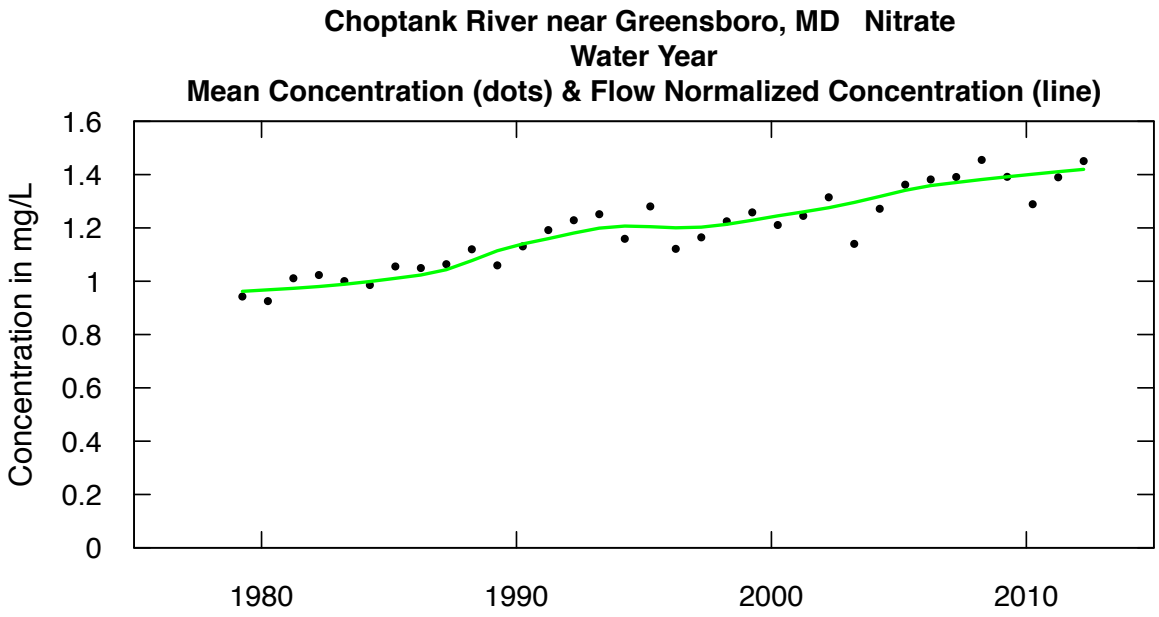


Figure 28. Concentration history graphic produced by the `plotConcHist` function for nitrate data for the Choptank River, Maryland.

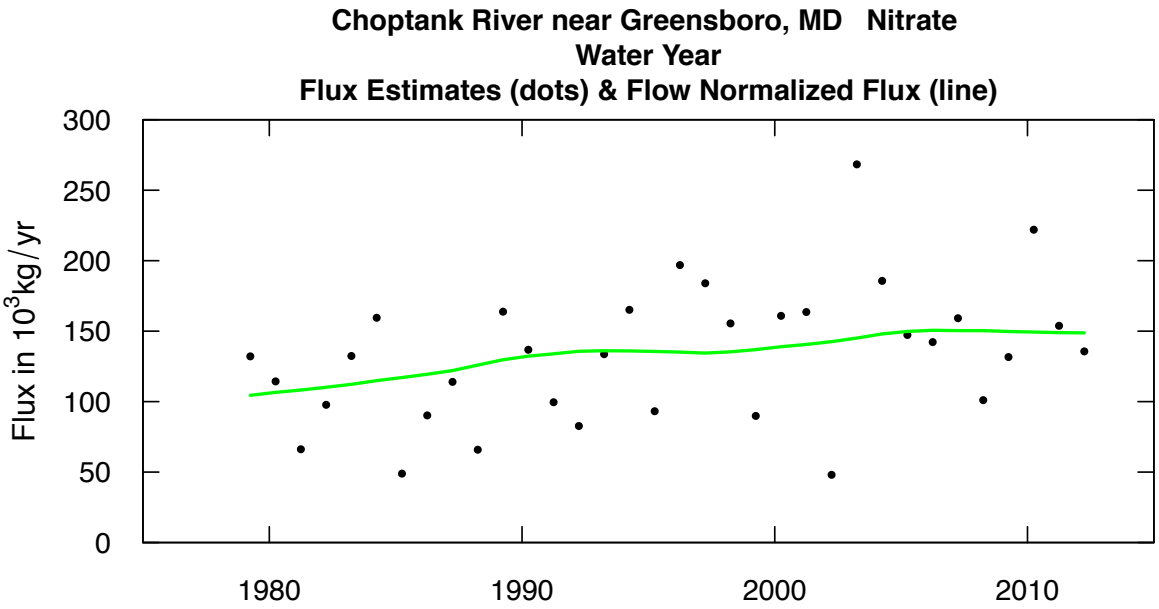


Figure 29. Flux history graphic produced by the `plotFluxHist` function for nitrate data for the Choptank River, Maryland.

`tableResults` has the following columns: year, mean discharge, annual mean estimated concentration, annual mean estimated flux, annual mean flow-normalized concentration, and annual mean flow-normalized flux. The user can select the units for discharge by using the `qUnit` argument (default is `qUnit=2`, which is m^3/s) and the units for flux by using the `fluxUnit` argument (default is `fluxUnit=9`, which is 10^6 kg/year). If a data frame designed for preparing a publication-quality table is desired, then set the argument `returnDataFrame` to `TRUE`. Here are two examples of the command:

```
tableResults(eList, qUnit = 3,fluxUnit = 6)
```

In this case, the user wants the discharge in 10^3 ft^3/s and flux in 10^3 tons/year with no data frame returned. If the data frame is desired, then the command might look like this:

```
resultsTable <- tableResults(eList, returnDataFrame = TRUE)
```

In this case, the user wants the default units (m^3/s) and 10^6 kg/year and wants a data frame named `resultsTable` to be returned. Figure 30 is an example of output for:

```
tableResults(eList, qUnit = 3, fluxUnit = 6)
```

```
Potomac River at Washington, DC
 Atrazine
 Water Year

 Year   Discharge    Conc    FN_Conc     Flux    FN_Flux
        10^3 cfs          mg/L           10^3 tons/yr

 1995      8.08    0.1331    0.1299     1.444     1.656
 1996     23.76    0.1718    0.1166     4.104     1.510
 1997     14.44    0.0926    0.1055     0.948     1.386
 1998     20.04    0.0930    0.0956     1.439     1.271
 1999      5.08    0.0716    0.0866     0.289     1.162
 2000      8.34    0.0737    0.0780     0.575     1.052
 2001      7.69    0.0671    0.0707     0.535     0.947
 2002      4.02    0.0568    0.0653     0.309     0.890
 2003     22.83    0.0922    0.0624     2.841     0.876
 2004     17.75    0.0660    0.0633     1.001     0.934
 2005     12.16    0.0594    0.0653     0.645     1.008
 2006      8.74    0.0642    0.0666     0.917     1.063
 2007     10.50    0.0482    0.0677     0.478     1.110
 2008     10.44    0.0699    0.0686     1.409     1.153
 2009      8.89    0.0751    0.0701     1.312     1.205
 2010     13.06    0.0500    0.0719     0.565     1.261
NULL
```

Figure 30. Example output from `tableResults` for atrazine data for the Potomac River at Washington, D.C.

Computing and Displaying Tables of Change Over Time

The function `tableChange` provides measures of change, in both flow-normalized concentrations and flow-normalized flux, between pairs of years selected by the user. The function computes a total of eight change measures for any given pair of years. For flow-normalized concentrations, the four change measures are: change in flow-normalized concentration, in mg/L; the slope of the change in flow-normalized concentration, in mg/L per year; change in flow-normalized concentration, in percent; and the slope of the change in flow-normalized concentration, in percent per year. For flux, the user specifies the units used to measure flux (the default units are 10^6 kg/yr). The four flux change measures are, respectively: change in flow-normalized flux; the slope of the change in the flow-normalized flux, in flux units per year (for example, 10^6 kg/yr/yr); the change in the flow-normalized flux, in percent; and finally, the slope of the change of the flow-normalized flux, in percent per year. The command is:

```
tableChange(eList, fluxUnit, yearPoints)
```

The object yearPoints is a vector of integer numbers, in ascending order, which is the full set of years for which the user wants to make comparisons. This is best explained by an example. If a user wants to examine the changes between 1996 and 2003 and between 2003 and 2010, then yearPoints is specified as:

```
yearPoints <- c(1996,2003,2010)
```

This indicates that the full set of comparisons (changes or slopes) to be made includes all of the possible ordered pairs of these years. If the requested set of yearPoints exceeds the actual data, the tableChange output reverts to a default set made up of 5-year increments and multiples of 5-year increments. In the case considered in this example, those ordered pairs are: 1996–2003, 1996–-2010, and 2003–10. When yearPoints are defined as shown above, the command for quantifying the changes by using flux units of 10^3 tons/year is:

```
tableChange(eList, fluxUnit=6, yearPoints)
```

```
  Potomac River at Washington, DC
  Atrazine
  Water Year

          Concentration trends
  time span       change     slope    change     slope
                    mg/L   mg/L/yr         %       %/yr

1996  to  2003    -0.054   -0.0077       -46       -6.6
1996  to  2010    -0.045   -0.0032       -38       -2.7
2003  to  2010    0.0095    0.0014        15        2.2


                Flux Trends
  time span          change        slope       change        slope
                10^3 tons/yr   10^3 tons/yr /yr      %         %/yr
1996  to  2003        -0.63       -0.091          -42          -6
1996  to  2010        -0.25       -0.018          -16        -1.2
2003  to  2010         0.39        0.055           44         6.3
```

Figure 31. Output from the tableChange function for atrazine in the Potomac River at Washington, D.C.

Or this could be done in a single command line as:

```
tableChange(eList, fluxUnit=6, yearPoints=c(1996,2003,2010))
```

The output for this command is shown in figure 31.

There is an alternative version of the function tableChange, called tableChangeSingle, which creates a data frame that contains the contents of the table. The arguments are the same as those in tableChange, except that there are two additional logical arguments, returnDataFrame and flux. If a data frame were going to be returned, then the command to produce only the concentration output portion of figure 31 would be:

```
changeTableConc <- tableChangeSingle(eList, yearPoints, returnDataFrame=TRUE)
```

To return only the flux results, the command would be:

```
changeTableFlux <- tableChangeSingle(eList, fluxUnit=6, yearPoints,
returnDataFrame=TRUE, flux=TRUE)
```

These two data frames, changeTableConc and changeTableFlux, can be written to files and used to create properly formatted tables for use in a report (see appendix 2, section 12).

A wide range of change measures are presented in the output of these functions to facilitate a variety of comparisons across constituents, sites, and time frames. In addition to comparing rates of change over various time periods, another comparison particularly worthy of note is of changes in flow-normalized concentration, in percent, to changes in flow-normalized flux, in percent, for the same time period. If the nature of the change in the system were such that the trend in the log of concentration was the same across the full range of discharges and the full range of seasons, then it is mathematically assured that the changes, in percent, for flow-normalized concentration and flow-normalized flux would be equal to each other. When the LOADEST model is used, this equality must hold true because it assumes that the trend in the log of concentration is the same across all discharges and across all seasons. In the example shown in figure 31, this is far from the case. The flow-normalized concentration change from 1993 to 2010 is estimated to be 13 percent, but the flow-normalized flux change over that same period is estimated to be 40 percent. This means that the change (in percentage terms) in concentration over this period of time is greater for high discharges than it is for low discharges.

It is entirely possible for the changes in flow-normalized concentration to be of the opposite sign from changes in flow-normalized flux. If, for example, there were large decreases in point source inputs of a pollutant but increases in nonpoint sources associated with high-flow events, then we would expect to see a negative trend in concentration but a positive trend in flux. For example, the total phosphorus record for the Susquehanna River at Conowingo, Maryland (Hirsch, 2012) shows that over the period 1982–2012, the change in flow-normalized concentration was a decrease of 17 percent, but the change in flow-normalized flux was an increase of 13 percent. The explanation for this apparent anomaly is that the monitoring site is at Conowingo Dam, just upstream of the Chesapeake Bay. This dam is becoming full of sediment, and a great deal of phosphorus is attached to those sediments. In recent years, high-flow events have delivered much higher fluxes of total phosphorus past the dam than they would have in previous years, because the reservoir's ability to store the sediment entering from upstream is greatly diminished from what it had been, and the tendency for scour of reservoir sediment has increased. These two factors contribute to a substantial increase in flux. At the same time, however, substantial efforts are being made upstream of the reservoir to limit phosphorus inputs from the Susquehanna River Basin, and concentrations of total phosphorus entering and exiting the reservoir at moderate to low-flow conditions have been declining. Because these improvements manifest themselves over a large fraction of the time, they lead to decreases in average total phosphorus concentrations in the water going past the dam, even though concentrations on the few highest flow days have been increasing. Thus, investigations of trends in concentration may not be at all informative about trends in flux. It is useful to recall that statements about average concentrations are really statements about concentrations integrated over time, but statements about average fluxes are really statements about the product of concentration and discharge integrated over time, and thus the days of the highest discharge can strongly influence flux trends and have little influence on concentration trends. An important attribute of the WRTDS method is that it allows the user to examine both concentration and flux within the same computational framework. These diverse kinds of changes, revealed by these comparisons of percentage changes in concentrations and flux, can be further explored by using some of the tools introduced in the later section on "Exploring Model Behavior and Adjusting Model Parameters."

Exploring the Quality of the Fitted Model—Overview

Even though the WRTDS model is highly flexible, there is no assurance that it will provide reasonable estimates of concentration and flux for all days in a record. The graphical tools can help identify serious problems with the model fit, and the EGRET package makes it easy to produce and view a large number of these graphics quickly. Many of these graphics are types of residuals plots, in which the model residuals (observed minus predicted values of log(c)) are plotted against predicted values or against other explanatory variables. The following is a list of the types of problems that these graphics can help identify.

- One of the problems that can arise with models such as WRTDS or LOADEST is a tendency toward severe underprediction or overprediction of concentrations on days of particularly high discharge. This can happen because the data at the highest discharges may be sparse and the model may not be sufficiently flexible to capture the particular curvature that exists in the ln(c) versus ln(Q) relationship. When this happens, it can result in severe underprediction or overprediction of annual or long-term mean fluxes.

- Another problem is the possibility that concentrations are severely underpredicted or overpredicted for a period of many months to years. This can arise particularly in large watersheds, where there can be long periods when the discharge is disproportionately derived from one portion of the watershed that may be a source of water with particularly high or particularly low concentrations. This can have a serious effect on annual or seasonal estimates of concentration or flux, but it is unlikely to have much of an influence on long-term mean flow-normalized values.

- Yet another type of problem can arise when the changing conditions of the watershed are dominated by abrupt changes in one particular point source. WRTDS makes an implicit assumption that changes in the system are gradual and typically a result of an aggregate of many actions taking place at different times. Examples include many small changes in

point source loadings, gradual shifts in land cover or in land-use practices across the entire watershed, gradual adoption of management practices by many landowners, or changes in farming practices or urban landscape modifications. There are certainly situations where a major cause of water-quality change is an enhancement of a single large point source discharge. Plots of residuals over time can help identify situations where this may be happening and can be used to design alternative methods to model properly what is happening (including downward adjustment of the `windowY` parameter).

- Finally, there are cases where the seasonality of the system is very strong and the changes between seasons are very abrupt. In such cases, the smooth representation of seasonality that WRTDS produces may not capture this pattern very well. Downward adjustment of the `windowS` parameter may help improve these results.

To facilitate rapid exploration of serious problems with WRTDS models, the EGRET software has a single function that produces a set of eight diagnostic graphics on a single page. This function is `fluxBiasMulti` and it is described in more detail below. The graphic it produces is designed to help the hydrologist quickly spot potential problems. Any one of the eight graphics in the plot can be produced individually in a larger graphic that would be suitable for publication or presentations.

Flux Bias Statistic

In the `fluxBiasMulti` function, the resulting graphic includes the flux bias statistic. This statistic was created in response to a problem that several researchers have identified regarding some of the more common methods for estimating average annual and long-term mean fluxes. This problem was first described by Stenback and others (2011), and that paper introduced the use of a statistic that is the functional equivalent of the flux bias statistic used here. Subsequently, others have described and explored this problem, notably Garrett (2012), Moyer and others (2012), Richards and others (2013), and (Hirsch, 2014). Most of the discussion in these papers has been about the use of the LOADEST model, which was first developed by Cohn and others (1992) and subsequently published in the LOADEST software package by Runkel and others, (2004). The LOADEST model is a regression-based method that uses either the same equation as WRTDS (equation 6) or one that also adds quadratic terms in ln(Q) and in time. LOADEST, unlike WRTDS, uses a single set of parameters to describe the relationship of ln(c) to discharge, time, and season, rather than using locally weighted regression like WRTDS. The motivation for development of the flux bias statistic was to use it as an indicator of possible flux bias in LOADEST models, but it can be used to explore flux bias in WRTDS models as well.

The flux bias statistic as defined here is a dimensionless representation of the difference between the sum of the estimated fluxes on all sampled days (P) and the sum of the true fluxes on all sampled days (O).

$$B = \left(P - O\right) / P \tag{13}$$

where:

$$O = \sum_{i=1}^{n} L_i = \sum_{i=1}^{n} k \bullet c_i \bullet Q_i \tag{14}$$

$$P = \sum_{i=1}^{n} \hat{L}_i = \sum_{i=1}^{n} k \bullet \hat{c}_i \bullet Q_i \tag{15}$$

where

L_i is the observed load on the i^{th} sampled day in kg/day

$\hat{L}_i$ is the estimated load on the i^{th} sampled day, in kg/day,

k is a units conversion factor = 86.4,

c_i is the measured concentration on the i^{th} sampled day, in mg/L,

$\hat{c}_i$ is the estimated concentration on the i^{th} sampled day; it is a "leave-one-out cross validation estimate" as discussed above in the section on the `modelEstimation` function,

Q_i is the discharge on the i^{th} sampled day, in m^3/s, and

n is the number of sampled days.

A value of *B* near zero suggests that the model is nearly unbiased. A positive value suggests a positive bias, and a negative value suggests a negative bias. Values of *B* that are between -0.1 and +0.1 indicate that the bias in estimates of the long-term mean flux is likely to be less than 10 percent. In a study (Hirsch, 2014) where true flux can be well estimated (because sampling frequencies are very high), it has been shown that the relationship between the true bias and the *B* statistic is highly nonlinear and rather imprecise. The flux bias statistic is not a basis for making corrections in flux estimates, but it can identify cases that are likely to have severe biases, which should motivate the hydrologist to seek some means to resolve that problem by use of a better statistical model. The work of Moyer and others (2012) and Hirsch (2014) indicates that WRTDS is substantially less prone to severe flux bias, but it is not immune to the problem. The flux bias statistic, shown on the `fluxBiasMulti` output, is one more tool that can help identify problematic situations where model parameters need to be adjusted or where the WRTDS model may simply not be suitable for application. Hirsch (2014) identified three common causes of severe flux bias: 1) a model that is poorly suited to the true relationship between ln(*c*) and ln(*Q*), 2) substantial seasonal differences in the shape or slope of the ln(*c*) versus ln(*Q*) relationship that are not accounted for by the model, and 3) substantial heteroscedasticity of model residuals. Identifying these three problems were the primary motivations behind the development of this set of graphics.

Graphics for Examining the Quality of the Model

As a rule, before any regression-based model is used, the hydrologist should carry out some graphical checks to determine if there are serious departures from the model assumptions, and then consider changing the model estimation method to account for those departures. Although the WRTDS model is designed to be highly flexible (that is, the assumptions it uses are less restrictive than other regression-based methods), it is still important to graphically evaluate the adequacy of the fitted model. The diagnostic tools shown here are tools for evaluating the appropriateness of such models. The design of the EGRET software makes it possible to check the adequacy of other models and compare them to WRTDS by creating alternative versions of the `Sample` and `Daily` data frames that contain the estimates from those alternative models. This is described in the section "Working with Multiple Versions of Data Frames."

The examples presented here come from a data set of 240 observations taken over a 10-year period for nitrate for the Vermilion River, at Pontiac, Illinois. The watershed is 1,500 km^2 at this location and is dominantly agricultural. Eight different graphics, are combined into a single graphic by using the function `fluxBiasMulti` (fig. 32). Each of these eight graphics can be used to produce a standalone figure, and more details about these standalone functions are in appendix 2 or the individual help pages for these functions. An additional graphic (fig. 33) is based on the same data set, but it was fitted with the half-window widths of the WRTDS model set to very high values (`windowY = 100, windowQ = 10, and windowS = 10`). By setting the half-window widths to such high values, the weights on all of the observations in each of the regressions become virtually equal. The consequence of this is that the model becomes essentially a nonweighted regression of the log of concentration on time, log discharge, and sine and cosine of time of year. This is virtually identical to the LOADEST-5 parameter model, which has often been used to fit these kinds of data sets for evaluation of fluxes and long-term water-quality trends. Several of the component graphics of figure 33 show some of the consequences that can arise from using an inappropriate model.

The arguments used in `fluxBiasMulti` that produced figure 32 are these (shown here with their default values): `qUnit = 2, fluxUnit = 3`, and `moreTitle = "WRTDS"`. The arguments `qUnit` and `fluxUnit` have been introduced in the section "Summarizing Water-Quality Data (without Using WRTDS)." The argument `moreTitle` allows the user to provide a name to the graphic, which will help to identify it in cases where multiple models are being considered. The default of `"WRTDS"` simply causes "WRTDS" to be printed as a part of the graphic's title; but any string of characters, enclosed in quotes, can be used here to identify the model, for example "approx LOADEST 5" (fig. 33). In the particular case shown here, the graphic in figure 32 was sent to a PDF of a particular size to assure the best possible formatting, although even without this formatting, the graphic is quite useable for identifying model errors. The set of three commands used for figure 32 were:

```
pdf("fluxBiasMultiVermNO3.pdf", height = 9, width = 8)

fluxBiasMulti(eList, fluxUnit=4)

dev.off()
```

This string of commands created a graphics file named `fluxBiasMultiVermNO3.pdf`, with a height of 9 inches and a width of 8 inches. The component parts of the graphic are these:

A. `plotResidPred` produces a graph of the residuals of the WRTDS model as a function of the model estimates. Both the residuals and the model estimates are in the natural log concentration units in which the model is fitted. The features of this graphic that are of interest are the degree of symmetry of the residuals around the zero line and the presence of curvature. Lack of symmetry would suggest that the errors of the WRTDS model depart from normality. Over many applications

of WRTDS, it has generally been found that modest amounts of asymmetry or departures from normality are generally not sufficiently important to require corrective steps be taken. Curvature is generally not a serious problem unless the smoothing window widths are much too wide for the range of values of the explanatory variables. The version of this graphic for "approx. LOADEST 5" shows a substantial amount of curvature. Of particular note is the fact that for estimates of log concentration greater than about 3, the residuals are all negative (fig. 33A), indicating that the model greatly overpredicts concentration. In addition, in a middle range of estimates, from about 1 to 2 log units, almost all residuals are positive, meaning that the model greatly underpredicts in this range.

B. `plotResidQ` produces a graph of the residuals of the WRTDS model as a function of the log of discharge (fig. 32B). In general, this figure will look rather similar to the previous one (fig. 32A), because discharge is often the single most important determinant of concentration in the WRTDS model. In this case, there is no apparent lack of symmetry or curvature that would suggest that the model is inappropriate. One feature that is particularly noteworthy here is that the variability of the residuals appears to be smaller at high discharges than at low discharges. This does not present a problem for the WRTDS model, because the variability is estimated for each observation or estimation point, and the bias correction is determined based on that estimate of variability. Some of the standard regression-based models (such as LOADEST) require an assumption of homoscedastic residuals, and even if those models properly dealt with the curvature of the data, they would not produce reasonable bias correction factors. The graphic in figure 33B shows again the strong curvature of the relationship, with almost all estimates for discharge values greater than about 50 m^3/s being too high (negative residuals). A particularly complex pattern of residuals at the lower discharge values shows positive residuals for discharges around 0.02 m^3/s, and negative residuals between about 0.1 and 1 m^3/s. This pattern suggests that the model used to fit these data is highly flawed.

C. `plotResidTime` produces a graph of the WRTDS residuals as a function of `DecYear` (fig. 32C). There are two specific issues that this type of figure might suggest, beyond those that the two previous figures might reveal. One is the possibility of a sudden step change at some point in time. Because WRTDS is fundamentally a smoothing algorithm, it is best suited to a situation where changes are gradual. For example, the changes may come from gradual adoption of new land-use management practices across many landowners and communities in a watershed, or the combination of a number of improvements in point-source controls, each one happening at a different time. However, there are certainly cases where a single event has a profound effect on water quality in a watershed. This might be the completion of a major upgrade of a large waste treatment facility, or a major change in the routing of wastewater or stormwater in a basin, perhaps through an interceptor sewer and into a different watershed or into a new holding facility for stormwater. If a step function is indicated in this plot, then that is an argument for doing other kinds of analyses that treat conditions before the change as a different population from those after the change, and then to evaluate the magnitude and nature of this shift. This might be done by creating two WRTDS models: one for the period before the shift and one for the period after the shift and comparing their behaviors. The other issue of interest that the `plotResidTime` graph may reveal is a relatively long period of dominantly negative or dominantly positive residuals, which can suggest that processes are at work in the watershed that deliver significantly different types of water to the monitoring location. One possible cause for this situation can be found in very large watersheds, where in one particular year a large fraction of the flow comes from one sub-watershed, while in other years that same sub-watershed supplies a small fraction of the flow. This kind of situation has been observed in the case of nitrate in the Missouri River (see discussion by Kahlkoff [2013]) during the flood of 2011. The source of this flood was heavy precipitation in the upper Missouri Basin, which has much lower nitrate concentrations than are produced from tributaries in the lower basin, which has been the source of most of the water in other recent flood events. Therefore, nitrate concentrations during this multiple month flood were all much lower than would be predicted by the WRTDS model. The results for the Vermillion River (fig. 33C) indicate that from the spring of 1993 through the fall of 1994, there was a long period of negative residuals. This departure started with the beginning of the extreme high flows of the 1993 flood, which affected most of the upper Mississippi River Basin including the Vermilion River, and continued until well after the conclusion of that event. This type of occurrence—that prolonged high flows can result in a subsequent period of negative residuals—is common, and it indicates that solute flux is lower than what would be expected based on discharge and time of year, because the large high-flow event had the effect of flushing the watershed of a significant part of its stored nitrate. The opposite condition is also true of low-flow events. High flows that follow long and severe low flows can have dominantly positive residuals because of the amount of solute that remains in storage in the watershed. This topic has been explored further in Murphy and others (2014). These kinds of long-term departures from residuals of approximately zero mean suggest that the error process is far from independent and that estimates for individual years could be improved by explicit use of the information seen in these residual time series. Development of enhancements of WRTDS to account for this kind of persistence is a high-priority topic for further model development.

D. `boxResidMonth` produces a graphic of the WRTDS residuals as boxplots by month (fig. 32D). In the ideal case, these boxes should all be approximately symmetrical around a value of zero. Substantial positive or negative departures for several months can indicate that the model is not fully accounting for real seasonal differences in system behavior. Some indication of this appears in figure 32D, where the residuals have a tendency to be positive particularly in the months of June, July, October, December, January, and February. The fact that these are spread across a wide range of months suggests that there is not a simple explanation for this departure and that there is no simple fix for it. In some cases, with large data sets, the use of a narrower seasonal window width (`windowS`) might be able to reduce some of this seasonal bias (it was not found to be useful in this case). Figure 33D shows a similar and somewhat stronger pattern, suggesting that the model approximating LOADEST-5 has less ability to fit the seasonal pattern than does the WRTDS model.

E. `boxConcThree` produces a graph containing three boxplots of concentration (fig. 32E). The first is a boxplot of the measured concentration values (`Sample$ConcAve`). The second is a boxplot of the WRTDS model estimates of concentration for all of the days on which there were measurements (`Sample$ConcHat`). As described previously, these are cross-validation estimates, and as such, they are computed without the knowledge of the actual concentration on the day for which they are estimated. The third boxplot is for the WRTDS estimates of concentration on all days in the period of record (`Daily$ConcDay`). The width of each of these boxplots is proportional to the square root of the sample size, thus the third box is substantially wider than the first two. Figure 32E is a good example of what we would expect to find for a model that is performing properly. In particular, the medians for all three sets are virtually identical, and the overall shapes of the distributions are similar in terms of the median and interquartile ranges. The distribution of actual sample values does show slightly more variability than the two for estimated values, in terms of the interquartile range and particularly the extreme values. This is exactly what should be expected. Estimates based on regression methods that fit the data reasonably well (such as WRTDS) will always show less variability than is seen in the sample data on which they are based. Estimates will always "regress to the mean." The fact that the second and third boxes are quite similar to each other is an indication that the set of sampled days covers a range of values of the explanatory variables that is similar to the range of values in the full set of days in the period of record. If the number of sampled days is rather small, or fails to cover a large part of the range of values for the full record, then the third box may show somewhat more variability than the second may. However, when there are serious problems with the model fit, (fig. 33E) for LOADEST-5, it is common for the extreme values of the sampled day estimates and of all daily estimates to be much greater than the extremes observed in the sample data. For example, in this case, the maximum sample value is 22.7 mg/L, the maximum estimate on a sampled day is 44.9 mg/L, and the maximum estimate on all days is 56.4 mg/L (in the WRTDS case the latter two were 16.5 mg/L and 16.9 mg/L respectively). High values of the daily estimates, which vastly exceed the high values in the original sample, are a very strong indicator that a model is seriously flawed. This plot provides a very simple way to make the necessary comparisons.

F. `plotConcPred` produces a scatter plot of observed concentration values (`Sample$ConcAve`) as a function of the estimates for those days (`Sample$ConcHat`) (fig. 32F, for WRTDS). It also shows a 1:1 line. A model that fits well should be tightly clustered around that 1:1 line and should generally be roughly symmetrical around that line. Average departures above the line should roughly balance departures below the line. This example shows some large departures from the 1:1 line, and some tendency towards underestimates when the estimated values are about 15 mg/L, but, conversely, some tendency towards large overestimates when the estimated values are between about 8 and 12 mg/L. It shows that errors can be large, although there does not appear to be a strong tendency towards bias in one direction or the other. Figure 33F provides another representation of the problem illustrated in the differences between the first two boxes in `boxConcThree`. It shows that estimates higher than about 20 mg/L are typically extreme overestimates, which can be anywhere from 5 to 30 mg/L higher than their true values. Conversely, for estimates between about 5 mg/L and 15 mg/L, there is a strong tendency for the model to produce underestimates of up to about 5 mg/L, but overestimates are infrequent and small. This graph is somewhat similar to what is shown in the `plotResidPred` graph (panel A of figs. 32–33),except that these values are shown in real concentration units and the bias correction factor has been applied here. Note that the residuals in panels A–D are residuals in the log of concentration and no bias correction is involved in computing these values. In contrast, panels F, G, and H all consider estimates of concentration or flux, and the computations of these values do involve use of the bias correction factor. Thus, these `plotConcPred` graphs show the compound effects of inadequate model fit and issues related to heteroscedasticity and its influence on the bias correction.

G. `boxQTwice` produces a pair of boxplots of discharge, on a log scale (figs. 32G and 33G). This graphic also appears in the output of `multiPlotDataOverview.` The first box represents the log discharge values on all sampled days (`Sample$LogQ`). The second box represents the discharge values on all days in the period of record (`Daily$LogQ`). The box plots are produced in the log units, which are then plotted against a logarithmic discharge scale (rather than being developed from the raw discharge data). These plots do not contain any information about the concentration data or the

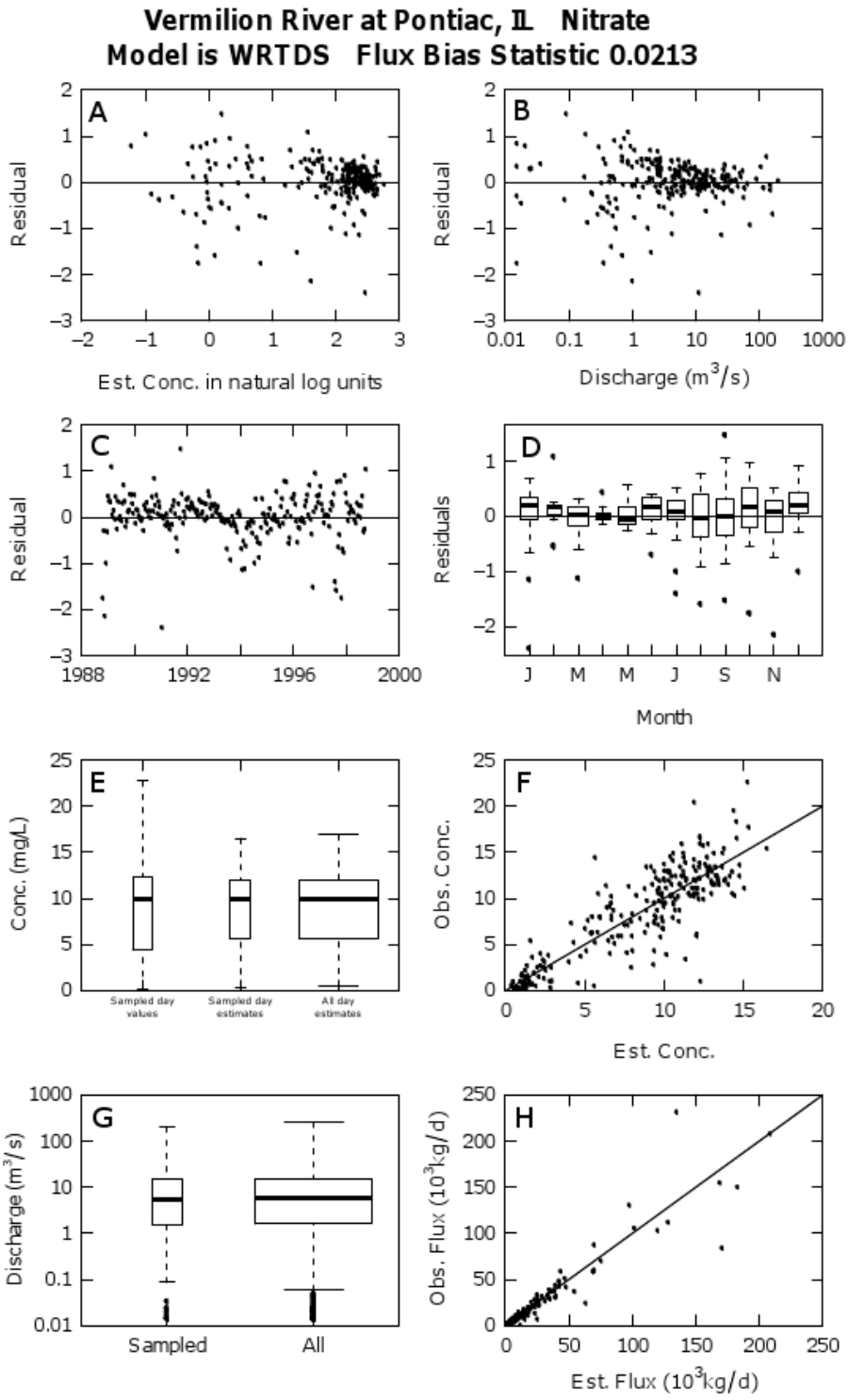


Figure 32. Output of the `fluxBiasMulti function` for nitrate for the Vermilion River at Pontiac, Illinois. (A description of each of the eight panels is presented in the text of the report).

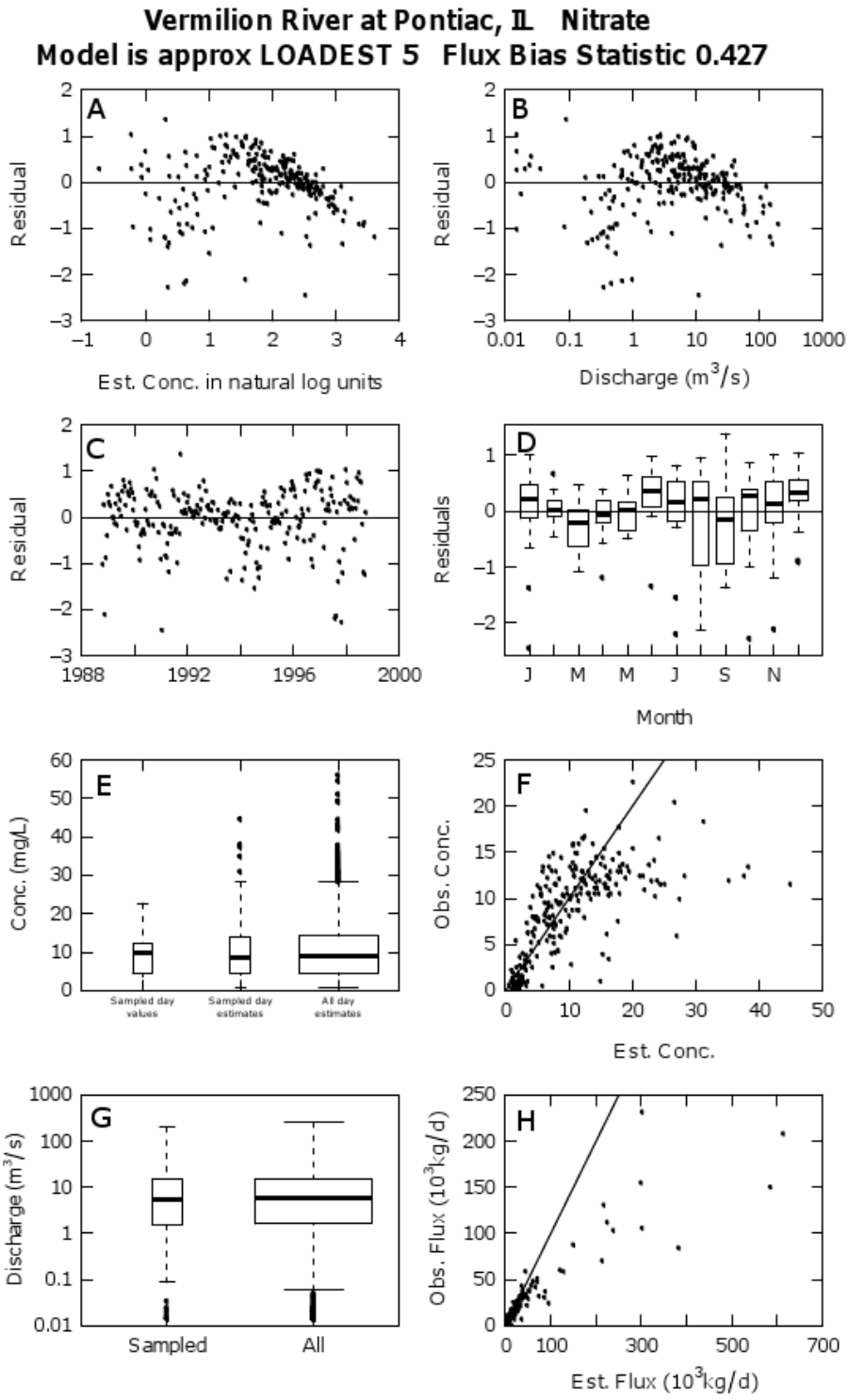


Figure 33. Output of the `fluxBiasMulti function` for nitrate (concentrations are in milligrams per liter, as N), Vermilion River at Pontiac, Illinois, using a model approximating LOADEST-5.

quality of the fit, but they are very useful indicators of the distribution of discharges in the sample data set. What would be particularly disturbing would be the case where the first box was shifted downwards in relationship to the second box. This would indicate a pattern of undersampling the higher discharges, which are often the most important to quantify, and this would be problematic for studies focused on the estimation of flux or trend in flux. Sampling schemes that preferentially sample at the higher discharges, but still cover much of the range of discharge values, have been shown to generally produce better estimates of flux than those that approximate a random sample of discharge values (see Hirsch, 2014). In this particular case, the two boxes are very nearly equivalent, which should come as no surprise, because the sample values are a random sample from a virtually complete daily data set.

H. `plotFluxPred` produces a scatter plot showing observed flux values on all sampled days (`Sample$ConcAve•Sample$Q•86.4`) as a function of estimated flux values on all sampled days (`Sample$ConcHat•Sample$Q•86.4`) (figs. 32H and 33H). It also shows a 1:1 line. The plot is designed to provide a graphical representation of potential bias in flux estimates. A biased model would show substantial asymmetry in the distribution of departures above and below the 1:1 line. In the case shown in figure 32H, the results are excellent and this is well reflected in the flux bias statistic shown at the top of the figure (0.0213, which represents an average error of +2.13 percent). In figure 33H, the estimates depart substantially from the 1:1 line. At the most extreme cases, days with estimated flux values of about $600•10^3$ kg/d, the true values were in the range of $150•10^3$ kg/d to $200•10^3$ kg/d. This strong bias is reflected in the flux bias statistic 0.427 shown at the top of figure 33 (an average error of +42.7 percent).

Two additional graphical functions can be useful for understanding quality of the model fit. These are `plotConcTimeDaily` and `plotFluxTimeDaily`. Both of these functions operate in the same manner, and this discussion will focus on `plotConcTimeDaily`. This function produces a graph of the daily estimates of concentration as a function of time (`Daily$ConcDay` as a function of `Daily$DecYear`), which is shown as a continuous line, and a set of points that represent the observed values of concentration as a function of time (`Sample$ConcAve` as a function of `Sample$DecYear`). Figure 34 is an example of a 3-year segment for the same data and model shown in figure 32.

The default command for this function is just `plotConcTimeDaily(eList),` and this will produce a plot for the entire period of the `Daily` data frame. In general, these plots will be somewhat difficult to interpret, because they can be dominated by the very steep rise and fall of the curve of estimated values. Making such a plot can help to identify time periods

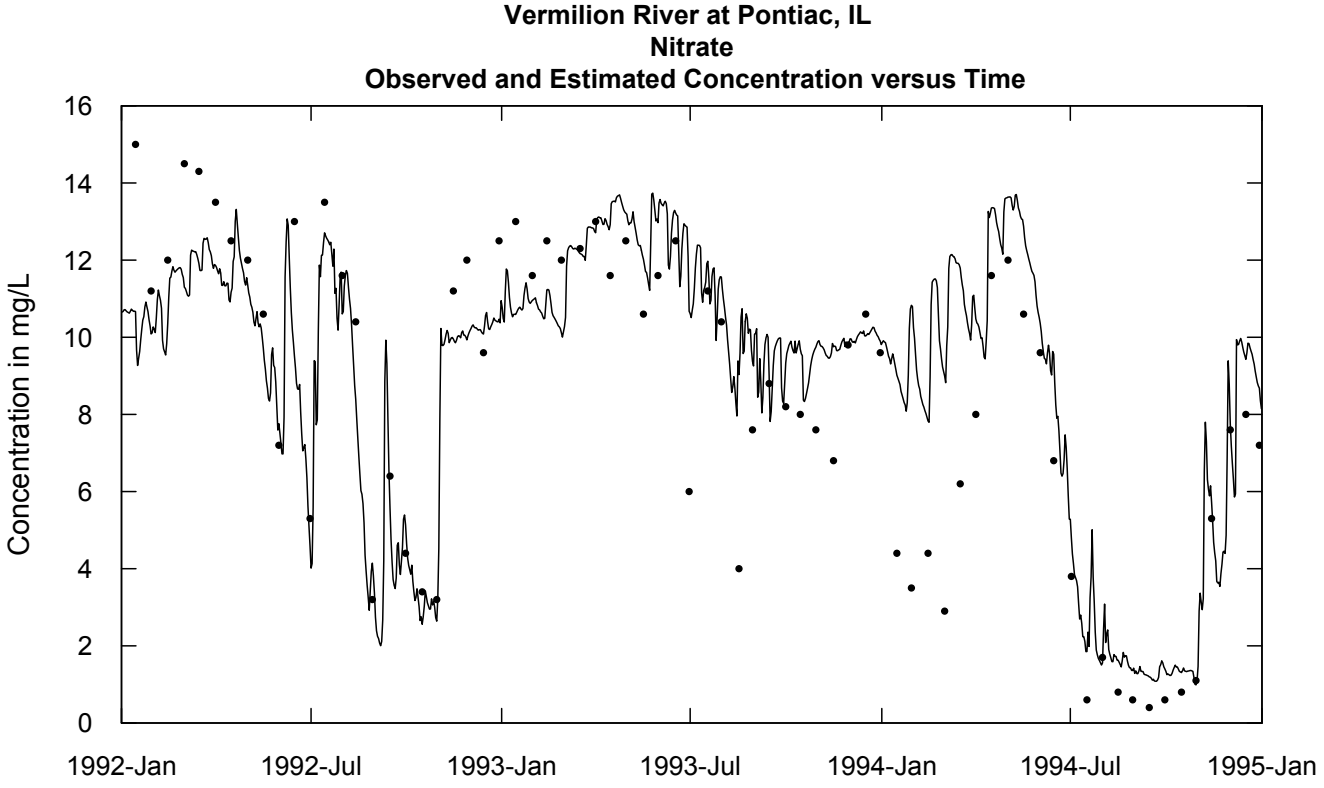


Figure 34. Output of the `plotConcTimeDaily` function for the years 1992–94 for the WRTDS model of nitrate concentration (milligrams per liter, as N) for the Vermilion River at Pontiac, Illinois.

within the record that may reveal interesting patterns, and once these periods have been identified, a revised version of the graphic can be produced that focuses on a short portion of the period of record. The second and third arguments to `plotConcTimeDaily` define the start and end of the time period to be plotted: `startYear` and `endYear`. Figure 34 was produced by the following command: `plotConcTimeDaily(eList, startYear = 1992, endYear = 1995)` or equivalently `plotConcTimeDaily(eList, 1992, 1995)`.

Several features of this figure are notable. For this period of the record, the model starts by doing rather poorly at estimating the very high values observed in the winter and early spring of 1992. The model then does well in capturing alternating high and low concentration values through much of the rest of 1992 and the prolonged period of relatively high values during the winter and spring of 1993. However, it does poorly at estimating concentrations during parts of the very high flow period of 1993 (summer) and very poorly in the winter and spring of 1994 (in the aftermath of the large flood), substantially overestimating concentrations during this period. In addition, the model severely overestimates the extremely low concentrations values in the late summer and early fall of 1994. This figure is a reminder that although the model fitted here has rather good properties in terms of overall fit and has a low flux bias, there can be extended periods for which it produces estimates that are either substantially too high or too low. There are important processes that influence concentration that this simple statistical representation does not capture, but doing the WRTDS analysis helps the user gain understanding of the system by removing the variability that can be explained by time, discharge, and season, thereby revealing the remaining variability. This kind of observation, of sustained overestimates or underestimates, can be an excellent point of departure for more detailed exploration of the factors that may be driving water quality at this location. This can be seen to some extent in panel C of figures 32 and 33, but figure 34, with its higher temporal resolution, may provide more insight about this behavior.

Figure 35 is a representation of the model estimates produced by the model used in figure 33 (the approximation of the LOADEST-5 model). Note that the vertical axis now runs from 1 to 40 mg/L, whereas in the figure 34 it ran only from 0 to 16 mg/L. As the previous diagnostic plots suggest, this model makes very large overestimates of concentration, particularly during the very wet spring of 1993 as well the winter and spring of 1994. It also seriously overestimates the very low concentrations in the summer of 1994, although it makes quite reasonable estimates during the late summer and fall of 1992 and 1993. In short, it shows concentration estimates that are much too high at both the extreme high flow and extreme low-flow conditions.

The function `plotFluxTimeDaily` operates in the same manner as `plotConcTimeDaily`. It is often more difficult to assess issues with the model by using `plotFluxTimeDaily`, because the range of variation in flux is typically so much

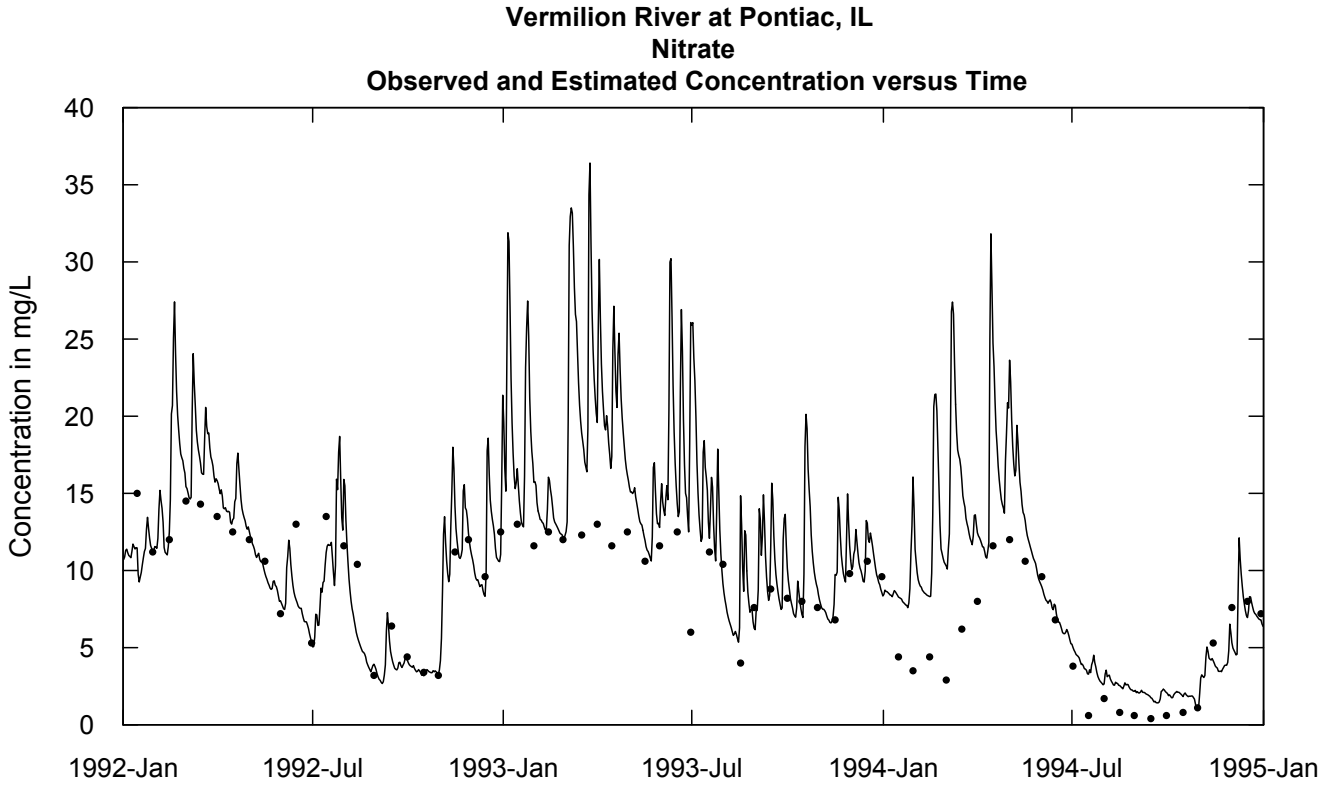


Figure 35. Output of the `plotConcTimeDaily` function for the years 1992–94 for the model that approximates the LOADEST-5 model for nitrate concentration (milligrams per liter, as N) for the Vermilion River at Pontiac, Illinois.

greater than with concentration. However, plotFluxTimeDaily can be useful for gaining a sense of how problems of model fit may influence flux results, which may be the topic of greatest interest in some studies.

Exploring Model Behavior and Adjusting Model Parameters

plotContours

The idea of contour plots of the three surfaces (yHat, SE, and ConcHat) was introduced in the section "WRTDS Analysis of Water-Quality Data." They are very central to the WRTDS modeling analysis, because they depict the model's full characterization of the behavior of concentration as a function of time, discharge, and season. This section is designed to provide guidance on how these graphics are generated and how they can be designed to be used effectively, both for exploration being done by the user and for final presentation of findings. The information shown in the graphics produced by plotContours comes from the object called surfaces which was created by the function modelEstimation. As such, all the choices regard window widths or the use of the edgeAdjust feature are all derived from the choices made when modelEstimation was executed.

A description of all of the possible arguments of the plotContours function is provided in the function help pages and in the vignettes, but this discussion will focus on the critical arguments that the user needs to use to obtain a useful result. These key arguments are listed in table 10. The user must consider a few basic choices when preparing one of these contour plots: setting the discharge range for the plot, deciding if flow-duration information will be shown on the plot, setting the time range for the plot, and setting the contour levels. The following paragraphs discuss each of these choices.

The vertical axis of the graphic is a discharge range that is determined by the two arguments qBottom and qTop. These arguments can be set to any values (provided that qBottom < qTop), but it is best that their values not extend to the most extreme high and low discharge values in the data set. The concentration estimates shown at the most extreme values of discharge will be much less reliable than those that are located towards the center of the distribution, and people tend to over-interpret the information depicted at these extremes. The estimates depicted at these extreme discharges will have very limited influence on the results, such as annual or long-term mean concentrations or fluxes, because they are in a low probability region of the discharge-time domain. Use of the flowDuration function before setting qBottom and qTop can be very helpful. A good rule of thumb is to set qBottom to a value close to the 5th percentile on the flow-duration curve and set qTop to a value close to the 95th percentile on the flow-duration curve. As with flowDuration, the plotContours function allows the user to select the units for depicting discharge (qUnit), and the values of qBottom and qTop are expressed in those units. For the best graphic results, the values of qBottom and qTop should be factor of 10 multiples of 1, 2, or 5 (for example 10, 20, 50, 100, 200, etc.).

One of the options is to have the plotContours graphic include a pair of curves superimposed on the plot that have a period of 1 year and represent seasonally specific flow-duration information. In the default case, they are plotted at the 0.05 and 0.95 points on the flow-duration curve, which is calculated from the daily flow information by using a moving seasonal window. The default width of this window is 60 days on either side of the center date. The advantage of plotting this flow-duration information is that it indicates those portions of the plot that depict relatively rare combinations of discharge and time of year and those that are centered in the midrange of the flow-duration curve. Because the plot is designed as a rectangle, portions of the plot can be based on very limited information (for example, very high flow in the dry season of the year) and be unimportant to the results because they represent conditions that are very rare. The disadvantage of presenting the flow-duration information is that it tends to complicate an already complicated and unfamiliar graphic. The user may want to include these curves to help define the best possible range of discharge values to use in a graphic, but when a graphic is produced for presentation, this information may be left out for the sake of simplicity. The argument used for plotting this curve is the logical variable, flowDuration. The default value is TRUE, meaning that these two curves are shown. A good rule of thumb is to make the vertical scale of the graphic such that the areas below the lower curve and above the upper curve are a rather small fraction of total plot area. This means that the lower plotting limit for discharge (qBottom) should be set somewhat higher than the minimums for the dry season of the year, and the upper plotting limit for discharge (qTop) should be set somewhat lower than the maximums for the wet season of the year.

The choice of time range for the plot is significant. Covering the entire period of record being used in the analysis is valuable from the standpoint of understanding the overall evolution of the system. The negative consequence is that, because of the seasonality that is typical of these surfaces, the vertical stripe effect becomes very pronounced, and changes, particularly for certain parts of the year, become difficult to observe visually. One approach to choosing a time range for the plot is to first produce a graphic that covers the entire period, and then, after determining what kinds of changes are of interest, produce another version of the plot that sets the starting and ending dates to cover fewer years (the first two arguments of the function are yearStart and yearEnd). One can produce a series of such graphics, each one covering a period such as 2 or 3 years, forming a progression of plots that could start 5 or 10 years apart in time.

Table 10. Arguments to the `plotContours` function.

Argument	Definition	Default
`eList`	Specifies the named list that contains the `INFO` and `surfaces` objects that will be used by this function.	No default, name of list must be stated.
`yearStart`	Starting date for the contour plot, in decimal years. Should be an integer value.	No default is established; the user must set this value.
`yearEnd`	Ending date for the contour plot, in decimal years. Should be an integer value.	No default is established; the user must set this value.
`qBottom`	The discharge value that forms the bottom of the plot, expressed in the units specified by `qUnits`. Because of the built-in log scaling, these should be values such as 1, 2, 5, 10, 20, 50, etc. The function `flowDuration` can be very helpful for specifying an appropriate value for `qBottom`. A good rule of thumb is to set the value slightly below the 5-percent level on the flow-duration curve.	No default is established; the user must set this value.
`qTop`	Discharge value that defines the top of the plot, expressed in the units specified by `qUnits`. Same issues as discussed above for `qBottom`. A good choice of values is slightly above the 95-percent level on the flow-duration curve.	No default value is defined; the user must select this value.
`whatSurface`	For `whatSurface` = 3, the plotted surface is the expected value of concentration (`ConcHat`). For `whatSurface` = 1, the plotted surface is the values of `yHat` (the expected value of log concentration). For `whatSurface` = 2, the plotted surface is the `SE` surface (the standard error of log concentration).	`whatSurface` = 3. Plot is the `ConcHat` surface.
`qUnit`	Determines the units to be used for discharge. See `printqUnitCheatSheet()` for a listing of discharge unit codes.	`qUnit` = 2. Discharge is in m3/s.
`contourLevels`	This is a vector of contour-level values, typically starting at zero and progressing in equal intervals to some maximum value. However, it need not go to zero and does not have to be equal interval. See the text below on how to specify the `contourLevels` argument.	`contourLevels` = NA. The contour levels are set by an automatic process. In most cases, this is a poor choice because the maximum is set too high, but it can be a good starting point.
`span`	This argument specifies the smoothness of the flow-duration curves plotted on the graph. It is the half window width for computing seasonal flow-duration levels.	`span` = 60. This is generally a good choice, and there is typically no need to adjust it.
`pval`	This is the probability value for the flow-frequency information shown on the plot. It specifies the probability of the lower curve; for example, `pval` = 0.05 specifies that the lower curve is the 5-percent level on the seasonal flow-duration curve. The upper curve is set at 1 – `pval`; for example, for `pval` = 0.05, the upper curve would be the 95-percent level.	`pval` = 0.05. This is generally a good choice.
`vert1`	Location of a vertical black line on the graph at a particular time specified by `vert1` (defined in decimal years). It is used to illustrate the idea of a "vertical slice" through the contour plot, which might be shown in a subsequent use of `plotConcQSmooth`.	`vert1` = NA. The result is that no vertical line is plotted.
`vert2`	Location of a second vertical black line on the graph.	`vert2` = NA. The result is that no vertical line is plotted.

Table 10. Arguments to the `plotContours` function.—Continued

Argument	Definition	Default
`horiz`	Location of a horizontal black line on the graph. It is expressed in the discharge units used in the plot. It can be used to illustrate the idea of a "horizontal slice" through the contour plot, which might be shown in a subsequent use of `plotConcTimeSmooth`.	`horiz` = NA. The result is that no horizontal line is plotted.
`flowDuration`	This is a logical variable. If TRUE the flow-duration lines are plotted on the graph. If FALSE, they are not plotted.	`flowDuration` = TRUE.
`color.palette`	This defines the color palette to be used for the contour levels in the plot; see appendix 2 for more information	color.palette=colorRampPalette(c("white","gray","blue","red"))

Setting the `contourLevels` variable is best done by using the R function `seq`. The `seq` function has three arguments: the first is the starting value in the sequence, the second is the maximum value for the sequence, and the third is the interval between the values in the sequence. Thus, if the desired contours were at 0, 5, 10, 15, 20, 25, and 30 mg/L, this would be indicated in by `contourLevels = seq(0, 30, 5)`. Figure 36 shows the resulting plot for the Vermilion River nitrate data discussed in the previous section. The command used to produce it is:

```
plotContours(eList, yearStart=1989,yearEnd=1998,qBottom=0.2,qTop=100,contourLeve
ls=seq(0,18,2))  or equivalently plotContours(eList, 1989,1998,0.2,100,contourLevels=
seq(0,18,2))
```

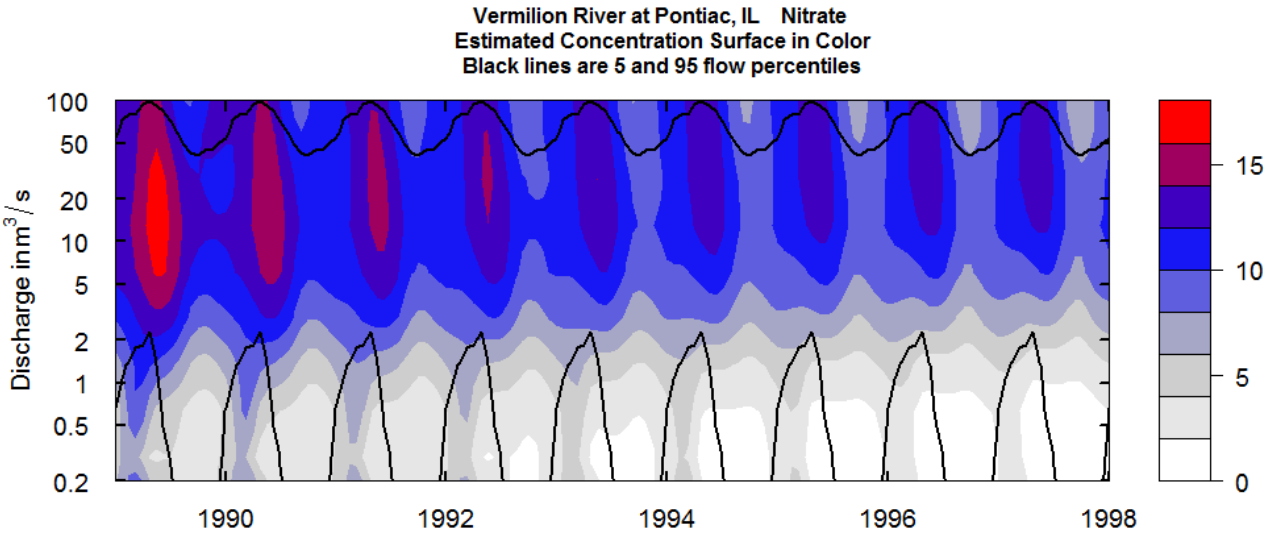


Figure 36. Contour plot output of the seq function for the WRTDS model of nitrate (in milligrams per liter, as N) for the Vermillion River at Pontiac, Illinois. Year designations on the horizontal axis indicate the start of the calendar year.

Selecting the best contour intervals is best done by a process of trial and error. The user can start with the default (where `contourLevels = NA`), but the total concentration range used to establish the colors for the contour intervals will probably be too large and will result in very poor differentiation of estimated concentrations at discharges and times of year that are of primary interest. The default levels will be too high because they are set by the maximum estimated concentrations in the entire Q versus `decYear` domain for the model, which includes some extreme and very low probability combinations of discharge and season. When the user makes an appropriate selection of `qBottom` and `qTop`, the maximum concentration estimates on the contour plot will often be much lower than the upper value of estimated concentration on the scale bar. By observing the highest estimates plotted when `contourLevels=NA`, the user can select a lower maximum value for the scale bar and a smaller

interval between contour levels than those automatically set by default. If the maximum value is not sufficiently high to cover the full range of values, the area that is in excess of the maximum will also be shown in white, and the maximum level should be increased to slightly exceed the maximum. The minimum value for the `contourLevels` can be greater than zero, although in many cases zero may be a good choice for the minimum.

Contour plots of the standard error of the WRTDS model can be made by using the argument `whatSurface = 2`. These plots can be useful for displaying one of the important reasons for using WRTDS rather than more standard regression models. Specifically, these plots show that SE can be very different across a range of time and discharge values. In standard regression methods, such as LOADEST, the assumption is that SE is constant. For example, figure 32 shows a great deal of heteroscedasticity of the residuals in the model whose concentration contours are depicted in figure 36. Figure 37 shows a 2-year example of the WRTDS estimate of the standard error of the model. The command to produce this figure is:

```
plotContours(eList, yearStart=1994,yearEnd=1996,qBottom=0.2,qTop=100,whatSurface=
2,contourLevels=seq(0.2,0.8,0.1))
```

As expected, based on the nature of the heteroscedasticity seen in this data set, the SE becomes substantially smaller as discharges increase above about 2 m^3/s, and the months of late summer and fall have the highest SE and the months of spring

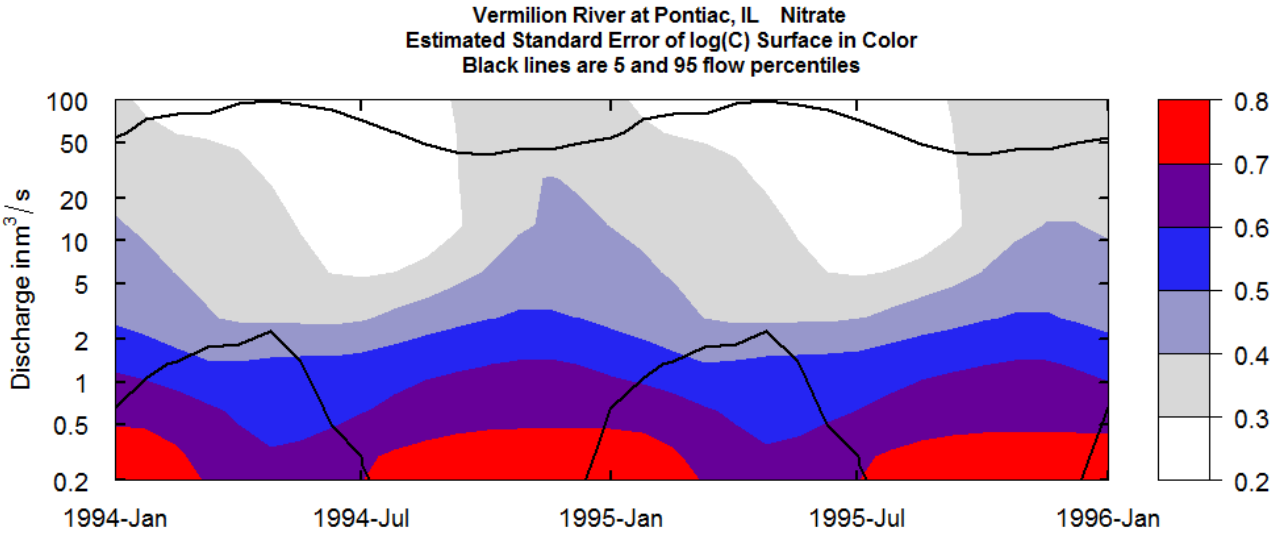


Figure 37. Contour plot output of the estimated standard error of log(concentration) for the WRTDS model of nitrate for the Vermillion River at Pontiac, Illinois.

have the lowest SE values. The importance of these variations is related to the computation of the BCF for the WRTDS model. As discussed above, the BCF is $\exp(SE^2/2)$. In a standard regression formulation such as the LOADEST models (Runkel and others, 2004), which assumes homoscedastic residuals, the BCF is nearly constant across the range of values of the explanatory variables (see Cohn and others [1992] for an explanation of the BCF for regression with homoscedastic residuals). In the case shown here, the BCF for the WRTDS model would range from 1.331 at very low discharges to 1.020 at very high discharges. A standard regression formulation would have BCF values that range only from 1.218 at very low discharges to 1.208 at very high discharges. Because the value of `ConcHat` is determined by multiplying exp(`yHat`) by the BCF, these differences in BCF values can be quite significant when estimates of concentration and flux are determined. In this example, at high discharges, which carry most of the flux, the WRTDS BCF is much smaller than the BCF for standard regressions. The result is that standard regressions will seriously overcorrect for bias, resulting in a large overestimate of flux. Conversely, at low discharge, the WRTDS BCF is much smaller than the BCF for standard regressions, and the result is that the standard regression approach will undercorrect at low discharges. However, these low discharges are relatively unimportant to total annual flux. The consequence is that standard regression approaches will overestimate flux because of their failure to consider heteroscedastic residuals. This issue is discussed in detail in Hirsch (2014).

Introducing an Example Case: Maumee River, Ohio, Dissolved Reactive Phosphorus

The example that will be used to further illustrate the use of `plotContours` and introduce the functions `plotDiffContours`, `plotConcQSmooth`, and `plotConcTimeSmooth` is that of dissolved reactive phosphorus (DRP) in the Maumee River at Waterville, Ohio. The data set used here is a subset of the very large data set of 38 years duration collected by the Heidelberg University National Center for Water Quality Research (*http://www.heidelberg.edu/academiclife/distinctive/ncwqr,* accessed November 2013). The full data set consists of 16,930 individual samples collected over the years 1974–77 and 1980–2012, but the data set used here is a randomly selected subset of 1,000 of these samples (an average of 27 samples per year). The data set used here is an interesting case because of the very substantial changes that have taken place over this long period of record and the fact that it illustrates a particularly interesting evolution of water-quality issues (Baker and Richards, 2002; Richards and Baker, 2002). Starting in the 1970s, DRP was very high and was particularly dominated by point sources, although some DRP came from nonpoint sources. The inputs from the Maumee and other major rivers were contributing DRP and other forms of phosphorus that were shown to be a major driver of the severe eutrophication of the western part of Lake Erie. Over a period of several years, substantial investments were made in treating these point sources, and therefore, DRP fluxes from the Maumee River into Lake Erie declined. Similar efforts at reducing point sources on other Lake Erie tributaries and the point sources that discharged directly to the lake also contributed to improvements in conditions in Lake Erie. However, in recent years, nonpoint sources of DRP have increased due to agricultural practices. These practices include broadcast application of fertilizer onto the surface, application in the fall rather than the spring— allowing both applied fertilizer and crop residue to remain very close to the land surface rather than being incorporated deeper into the soil profile—increased tile drainage, and soil compaction due to heavy equipment. The net effect of these practices is creation of a surface layer of soil with excess phosphorus, which is readily mobilized in the dissolved phase during times of high surface or subsurface flow. Consequently, conditions in Lake Erie have deteriorated, and in the summertime the lake is now severely impaired by algal blooms (especially blue-green algae, some of which are toxic) and hypoxic conditions. (Michalak and others, 2013) (International Joint Commission, 2014). The following graphics illustrate the particular tools in WRTDS that can help to elucidate the nature of the long-term changes in DRP flux in the Maumee River.

Because the Maumee River record is quite long and the monitoring period contains a substantial break (little or no data in 1978, 1979 and 1980), a useful approach to visualizing the change is to use `plotContours` to consider the relation of concentration to discharge and time of year for three different periods in the record. The first is 1975–77, the second is 1988–89, and the third is 2010–11. The command used to create the contour plot for the first of these periods is:

```
    plotContours(eList, yearStart=1975, yearEnd=1977, qBottom = 20,qTop = 500, con-
tourLevels = seq(0,0.28,0.04))
```

For comparison, this plot was combined with the other two time periods to form figure 38. The contents of the other two contour plots would be obtained by changing the two arguments (`yearStart` and `yearEnd`) to 1988 and 1990 in the second plot and 2010 and 2012 in the third plot.

The top panel, from the mid-1970s, shows the highest concentrations taking place in the winter months at all flows, but high concentrations are most pronounced at the lowest discharges. This pattern of decreasing concentration with increasing discharge suggests the importance of point source inputs. Concentrations are also relatively high, almost without regard to discharge, during the late summer and early fall, when plant uptake of phosphorus from the soil is declining and the harvest of crops and leaf fall creates a large reservoir of available phosphorus on the landscape. The situation is quite different in the late 1980s. Overall, the concentrations are much lower, there is little indication of major point source inputs, and the highest concentrations of P are during the months of July through January. Finally, the most recent period is similar in nature to the late 1980s but the concentrations at higher discharges are higher than they were in the mid-1980s at all times of the year.

Another way to view the changes that have taken place over some period of time is to use the function `plotDiffContours`. This function computes and plots the difference between the contoured surfaces for any selected pair of years in the record. The arguments to this function are largely the same as those in `plotContours`, but instead of the arguments `yearStart` and `yearEnd`, this function uses `year0` and `year1`. The only other difference is that there is no `contourLevels` argument, but in its place is an argument called `maxDiff`. This argument controls the range of contour intervals used in the graphic. The red colors depict the increases over the selected period, and the maximum increase they will depict is equal to `maxDiff`. The blue colors depict the decreases over the selected period and they range from zero to `maxDiff`. Any region of the contour plot that has a difference that is greater than `maxDiff` in absolute value will be shown in white. If there is an area of white, the user should increase `maxDiff` enough so that the graphic contains no white area. The following example, shown in figure 39, explores the changes from 1988 to 2011. The command used is:

```
    plotDiffContours(eList, year0 = 1988, year1 = 2011, qBottom = 20, qTop = 500,
maxDiff=0.12)
```

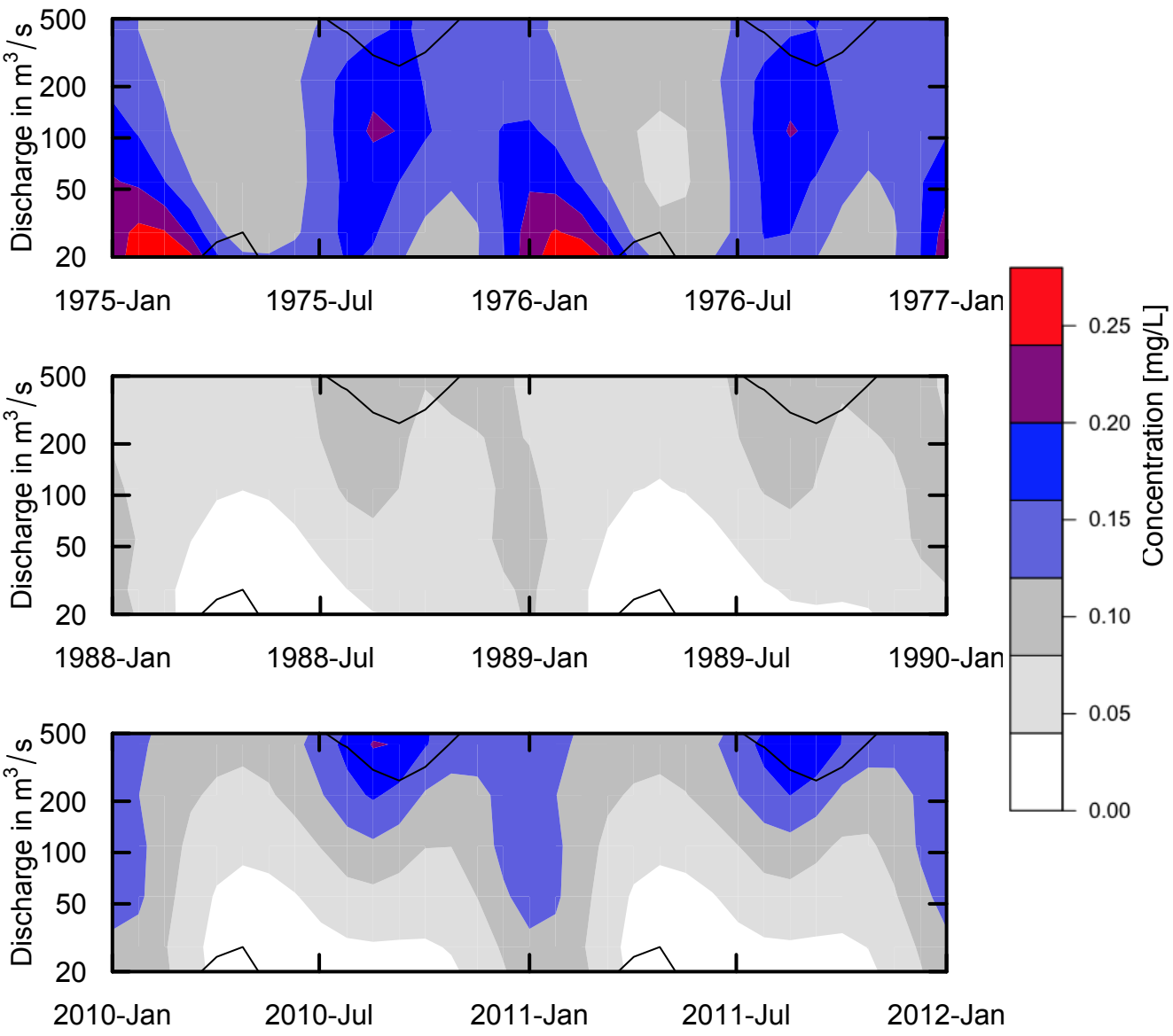


Figure 38. Contour plot output of dissolved reactive phosphorus for the Maumee River at Waterville, Ohio, for three 2-year time periods.

Figure 39 shows that average concentrations have increased at all discharges during the late fall and through the winter, which is after the fall fertilizer application period and before crop uptake of phosphorus has begun for the growing season. The increases tend to be larger at the higher discharges than at lower discharges. They appear to be largest in the late summer and early fall at the highest discharges; however, it should be noted that the conditions that give rise to the largest increases shown on the figure are highly unusual. They lie above the 95th percentile of the flow-duration curve for that part of the year, so these estimates are likely based on relatively few observations in this range and have very limited influence on average annual concentrations or flux, because they represent conditions that happen with low frequency. Another item to note from this figure is that there has been very little change in concentration at moderate to low discharges from about April through November and even a small decrease at low discharges in the late summer and fall.

plotConcQSmooth

An alternative way to view these surfaces is to plot "slices" through them, and the function `plotConcQSmooth` produces vertical "slices" through the contour plot. It plots the WRTDS estimate of concentration as a function of discharge, and it can do so for as many as three different points in time. The arguments to `plotConcQSmooth` are listed and described in table 11.

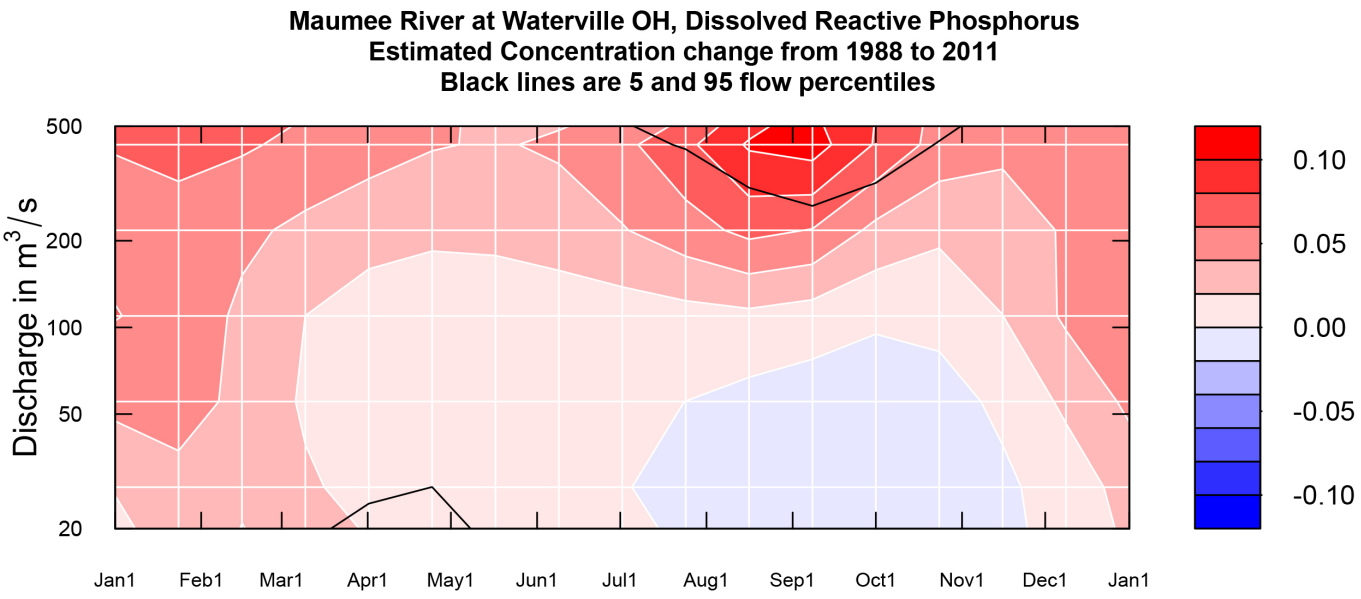


Figure 39. Difference contours produced by `plotDiffContours` for dissolved reactive phosphorus for the Maumee River at Waterville, Ohio, for the period from 1988 to 2011.

Table 11. Arguments for the function `plotConcQSmooth`.—Continued

Argument	Definition	Default
`eList`	Specifies the named list that contains the `INFO` and `Sample` data frames that will be used by this function.	No default, name of list must be stated.
`date1`	A date, in the form "yyyy-mm-dd" (quotes must be used), specifying a date around which the concentration estimate is to be made using the WRTDS model.	No default is established, the user must set this value.
`date2`	A second date, in the form "yyyy-mm-dd," to be used for the second curve on the figure. If no second curve is wanted, then `date2=NA`.	No default is established, the user must set this value.
`date3`	A third date, in the form "yyyy-mm-dd," to be used for the third curve on the figure. If no third curve is wanted, then `date3=NA`.	No default is established, the user must set this value.
`qLow`	Lowest discharge to be used on the figure, expressed in the units specified by `qUnits`. See the text below for a discussion of selecting appropriate `qLow` and `qHigh` values.	No default is established, user must set this value
`qHigh`	Highest discharge to be used on figure, expressed in the units specified by `qUnits`. See text below for a discussion of selecting appropriate `qLow` and `qHigh` values.	No default is established, user must set this value.
`qUnit`	Determines the units to be used for discharge. See printqUnitCheatSheet() for listing of discharge unit codes.	`qUnit` = 2. Discharge is in m^3/s.
`legendLeft`	The location of the left edge of the legend, expressed in the discharge units used in figure.	`legendLeft` = 0 (this allows the function to set `legendLeft` automatically).
`legendTop`	The location of the top edge of the legend, expressed in concentration units (mg/L).	`legendTop` = 0 (this allows the function to set `legendTop` automatically).
`printLegend`	A logical variable, if TRUE legend is included, if FALSE, legend is not included	printLegend = TRUE

Table 11. Arguments for the function `plotConcQSmooth`.—Continued

Argument	Definition	Default
`concMin`	Lower bound on the vertical axis for the plot. This will be ignored if the vertical scale is arithmetic (`logScale = FALSE`), in which case the lower bound is always 0 mg/L. This can be useful when multiple plots are being compared.	`concMin = NA`, which results in the program selecting the lower bound based on the curves being shown (if `logScale = TRUE`), and results in a lower bound of 0 if `logScale = FALSE`.
`bw`	A logical variable. If `bw = FALSE`, curves are plotted in color. If `bw = TRUE`, curves are plotted in black and white.	bw = FALSE
`printTitle`	A logical variable. If `printTitle = TRUE`, a title is printed. If `printTitle = FALSE`, a title is not printed.	printTitle = TRUE
`printValues`	A logical variable. If `printValues = TRUE`, in addition to plotting the graphic, the function prints the values plotted to the console and can create a data frame with these values (see discussion below).	printValues = FALSE
`windowY`	Half-window width for time, in years, for the WRTDS smoothing method (see notes on selection of smoothing parameters below).	windowY = 7
`windowQ`	Half-window width for discharge, in natural log units, for the WRTDS smoothing method (see notes on selection of smoothing parameters below).	windowQ = 2
`windowS`	Half-window width for seasons, in years, for the WRTDS smoothing method (see notes on selection of smoothing parameters below).	windowS = 0.5
`minNumObs`	Minimum number of observations required to run a specific weighted regression (see notes on selection of smoothing parameters below).	minNumObs = 100
`minNumUncen`	Minimum number of uncensored observations required to run a specific weighted regression (see notes on selection of smoothing parameters below).	minNumUncen = 50
`logScale`	A logical variable. If `logScale = TRUE`, vertical scale is a log scale. If `logScale = FALSE`, vertical scale is arithmetic and starts at 0 mg/L.	logScale = FALSE
`edgeAdjust`	A logical variable. If `TRUE` the half window width for time weighting is increased when the estimation time is less than `windowY` years from the beginning or end of the estimation period. It is adjusted so that the full width of the window is equal to 2*`windowY`. If FALSE, the window width is not adjusted near the beginning or end of the data set. If FALSE, the optimal `windowY` value may be closer to 10 rather than 7 years.	edgeAdjust = TRUE

The selection of values of `qLow` and `qHigh` is facilitated by using the `flowDuration` function. In general, it is best to have the graph cover a range of discharge values that runs from about the 10- to 90-percent levels on the flow-duration curve that is specific to the time of year that will be portrayed in the graph. For example, if the curves are all set up for August 1 of three different years, then the appropriate command to use to identify the `qLow` and `qHigh` would be `flowDuration(eList, centerDate="08-01",span=60)`. This command calls for a flow-duration curve specific to dates that are plus or minus 60 days from August 1 of each year of the record, and the flow values will be reported in units of m^3/s (the default for `qUnit` in this function is 2). If other units will be used in the `plotConcQSmooth` function, then those units should be specified in the call to `flowDuration` by using the `qUnit` argument. If the three dates being used in `plotConcQSmooth` are for different times of the year, then the choice of `qLow` and `qHigh` should be based on the full year flow-duration curve, so the command should be `flowDuration(eList)` (with the `qUnit` value specified if it is other than m^3/s). In this situation, it may be better to use a discharge between the 10- and 25-percent levels for `qLow` and between the 75- and 90-percent levels for `qHigh`.

One example of the use of this function would be to produce a set of three curves depicting the concentration versus discharge relation in three different years, but all at the same time of year (so the variability due to season is eliminated). The command:

```
plotConcQSmooth(eList, date1="1975-08-01",date2="1988-08-01",date3="2010-08-
01",qLow=10,qHigh=300, logScale=TRUE,legendLeft=100,legendTop=0.06,printTitle=FALSE)
```

produced the plot is shown in figure 40. In this case, the three dates chosen were August 1 of the first year shown in each of the contour plots in figure 38. The range of discharge selected for the plot runs from 10 m^3/s to 300 m^3/s, which is approximately the 10- to 90-percent range on the flow-duration curve for conditions centered on August 1 with a span of 60 days. Setting the range of discharge values for `plotConcQSmooth` too wide creates the risk that the two ends of each curve will not be particularly meaningful, because they are too heavily influenced by just a few of the highest or lowest discharge sample values. The `legendLeft` and `legendTop` values were set after an initial plot was generated without specifying these arguments. The default location of the legend superimposed it over two of the curves, so these values were set to assure that the legend is in a portion of the figure without any curves. The concentration values are plotted on a log scale (`logScale=TRUE`).

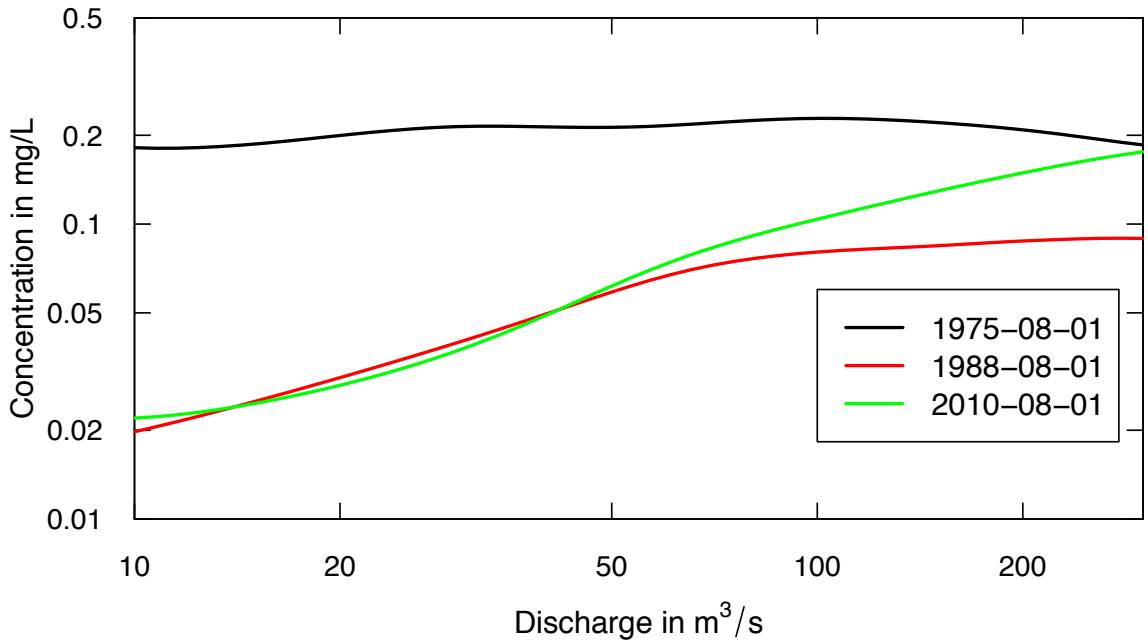


Figure 40. Plot produced by the `plotConcQSmooth` function for the relation between concentration of dissolved reactive phosphorus and discharge, centered on August 1 for three different years for the Maumee River at Waterville, Ohio.

This graphic depicts a very substantial change in the behavior of this watershed over time. Running another version of the same command provides an easy way to compare the discharge and concentration values:

```
augResults<-plotConcQSmooth(eList, "1975-08-01","1988-08-01","2010-08-01",10,300,
printValues=TRUE,logScale=TRUE,legendLeft=100,legendTop=0.06,printTitle=FALSE)
```

This command saves an object, named `augResults` (so named to be suggestive of august results), which is a table of the discharge and concentration values that make up the three curves (they are a set of 48 discharge values, equally spaced on the log discharge scale between 10 and 300 m^3/s). Then simply typing the command `augResults` prints this table so that comparisons can easily be made.

What is seen here is that in the early part of the record around 1975, estimated concentrations were relatively constant over the range of discharges, and the highest value is only about 23 percent higher than the lowest value. The shape of the curve suggests that DRP is derived from a mixture of a constant point source and a nonpoint source that is highly related to discharge.

Thirteen years later, around 1988, the picture is radically different. The highest values of concentration take place at the highest discharges, and these concentrations are about 350 percent larger than the concentrations seen at the lowest discharge. It indicates a decrease in concentrations of about 89 percent at discharges around 10 m^3/s and a decrease of only about 52 percent at the highest discharges. This suggests a very substantial decrease in the point source loading of DRP and a modest reduction in the nonpoint source loading. Then, moving from 1988 to 2010, the figure suggests that for low and moderate discharges, there is very little change in the system over these 22 years, but at discharges above about 100 m^3/s, the more recent period had substantially higher concentrations. For example, at a discharge of 300 m^3/s, the increase is about 96 percent over this period. This indicates that the point source controls seem to continue to be stable and effective, but the nonpoint sources that drive the DRP concentrations at higher discharges are continuing to increase. So, even though concentrations at low flow declined over time and have stayed low, at high flow they appear to have increased a good deal in the later years of this record.

Based on these observations, an obvious next question is: Does the general pattern change if a different time of year is considered? For example, in the following command, the dates are shifted from August 1 to May 1. The command that produced figure 41 is:

```
mayResults<-plotConcQSmooth(eList, "1975-05-01","1988-05-01","2010-05-01", 40,
700, legendLeft=200,legendTop=0.04,printValues=TRUE,logScale=TRUE,printTitle=FALSE)
```

There are differences in details between this May plot and the August plot in figure 40, but the overall message is the same: the concentration versus discharge curve has changed from one that is relatively flat over the relevant range of discharges in 1975, to one in 1988 where the concentrations at low discharges decreased by about 78 percent and concentrations at high discharges also decreased, but only by about 56 percent. Over time, however, the concentrations at high discharges have risen such that by 2010 they are about 78 percent of the high values from 1975. The patterns seen in these two figures provide a very strong case for using WRTDS as a way of describing the evolution of this system, compared to using LOADEST, which requires that the shape of the log concentration to log discharge relationship, for any given time of year, must remain the same over the entire period of record. These figures indicate that this relation can dramatically change shape over a period of multiple decades. Use of the `tableChange` function shows that the change in concentration over the period 1988 through 2010 was an increase of 43 percent, but the change in flux for the same period was 91 percent. This is an excellent example of why analysis of concentration trends is not necessarily meaningful for describing trends in flux.

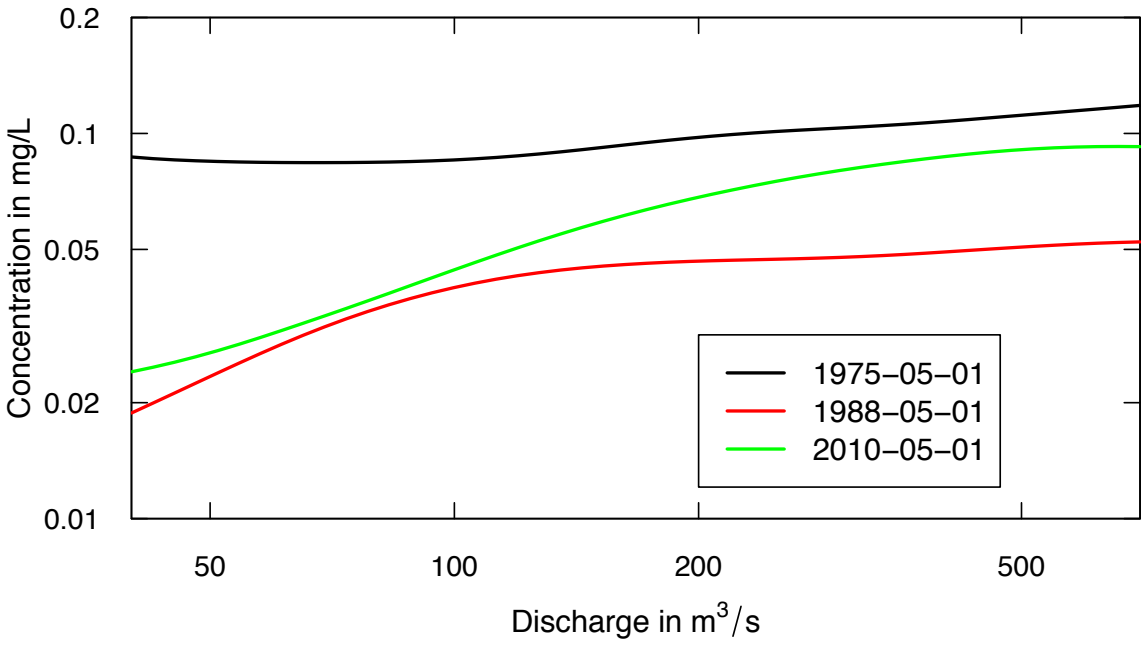


Figure 41. Plot produced by the `plotConcQSmooth` function for the relation between concentration of dissolved reactive phosphorus and discharge, centered on May 1 for three different years for the Maumee River at Waterville, Ohio.

Yet another type of comparison is possible with the `plotConcQSmooth` function, and that is the comparison across different seasons. Figure 42 was produced by using the following command:

```
plotConcQSmooth(eList, "2010-01-01","2010-05-01","2010-09-01",30,300,logScale=TRUE,l
egendLeft=150,legendTop=0.04,printTitle=FALSE)
```

When the year is held constant, the `plotConcQSmooth` function simply considers how the relation between concentration and discharge changes across the seasons, in this case exemplified by the curves for January 1, May 1, and September 1. Figure 42 shows that the January curve is nearly horizontal (no change in concentration as a function of discharge), but the curves for May and September rise steeply (increasing by a factor of 4 or 5 over the order of magnitude change in discharge). At lower discharges, the May concentration values are the lowest of the three, reflecting the active plant uptake of phosphorus by plants at this time of year and that the high levels of phosphorus that were available in January have been substantially depleted by May. September values are intermediate and show the likely influence of the end of the growing season, when the phosphorus content of the plants is becoming available for dissolved transport. Finally, the winter low flows show the effect of minimal plant uptake, the recent fall fertilizer applications, and the large amount of dead plant material on the landscape. At higher discharges, the concentrations are high in both September and January and somewhat lower in May (associated with plant uptake during the growing season). These results are in sharp contrast to the results that would be produced by a LOADEST model. LOADEST requires that the log concentration versus log discharge relation be the same shape and slope across all seasons of the year. These three curves from the WRTDS model suggest that this may be a very poor model assumption in this particular case.

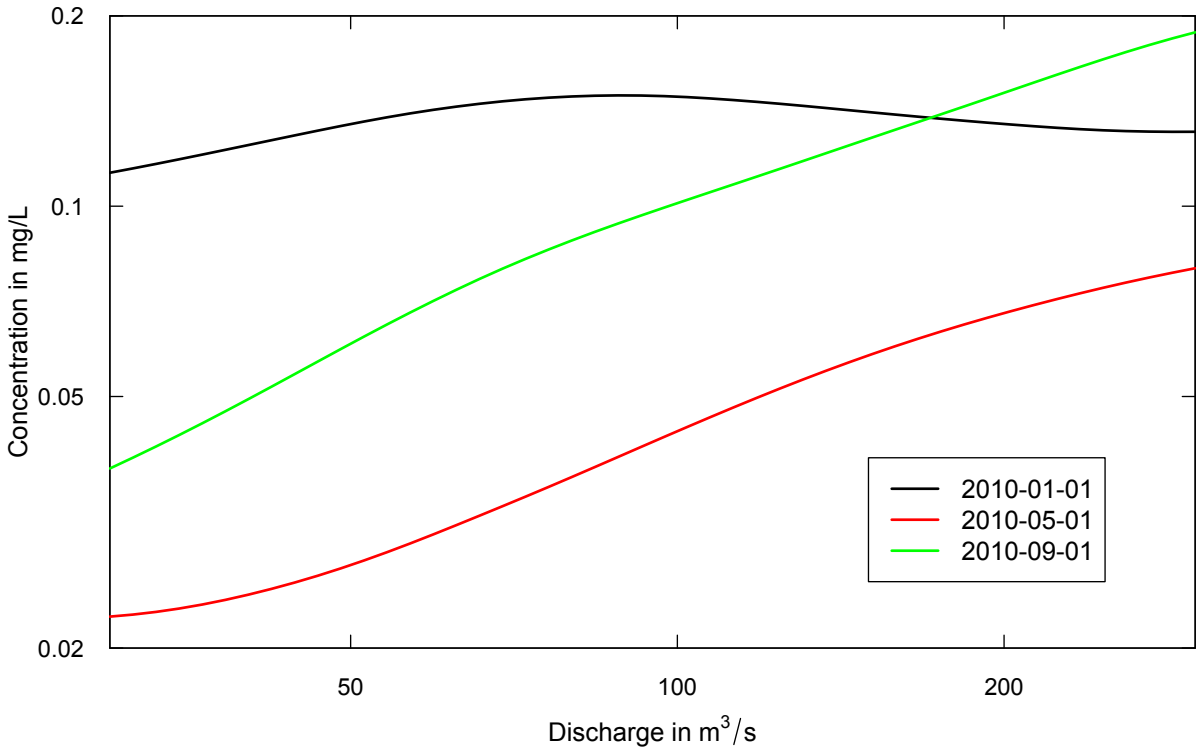


Figure 42. Plot produced by the `plotConcQSmooth` function for the relation between concentration of dissolved reactive phosphorus and discharge centered on three dates (January 1, May 1, and September 1) during 2010 for the Maumee River at Waterville, Ohio.

One additional type of use for the `plotConcQSmooth` function allows for experimentation with the setting of the WRTDS smoothing parameters. In particular, the user may be interested in using a smaller window for discharge (`windowQ`). An alternative version of figure 42 might use windowQ = 1 (rather than the default value of 2). Figure 43 was created with the command:

```
plotConcQSmooth(eList, date1="2010-01-01",date2="2010-05-01",date3="2010-09-01",qBo
ttom=30,qTop=300,logScale=TRUE,legendLeft=150,legendTop=0.04,printTitle=FALSE,
windowQ=1)
```

Figure 43 is not dramatically different from figure 42, but subjectively the original (fig. 42 with `windowQ = 2`) seems more plausible. The slightly wavy curves shown in figure 43 seem to invite an overly detailed interpretation of the data. The selection of "optimal" window widths remains a research challenge for the WRTDS method, but extensive experimentation has indicated that the default values provide results that are about as accurate as the method can obtain. As described in Hirsch and others (2010) and Sprague and others (2011), there may be situations on very large rivers (for example, the Mississippi or Missouri Rivers) where `windowQ = 1` or `1.5` may perform better than `windowQ = 2`, but this does not appear to be the case for this example.

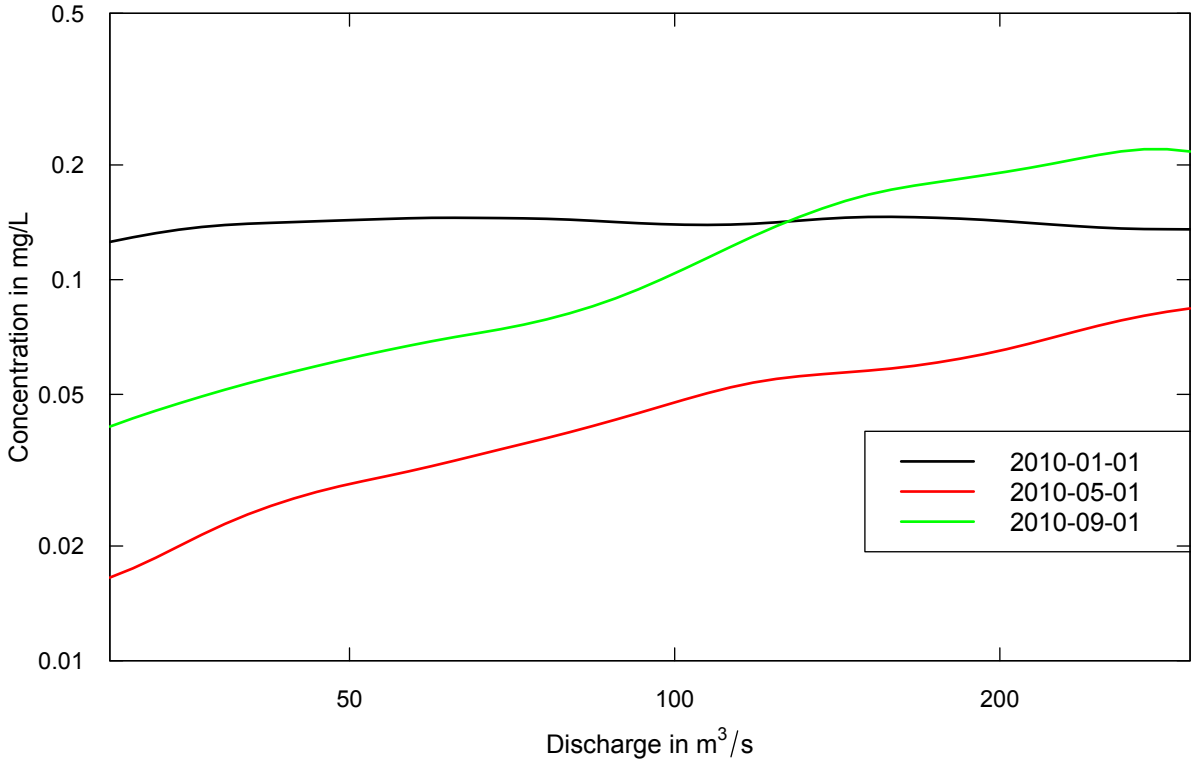


Figure 43. Plot produced with the discharge smoothing parameter (windowQ = 1, rather than windowQ = 2) for the relation between concentration of dissolved reactive phosphorus and discharge centered on three dates (January 1, May 1, and September 1) during 2010 for the Maumee River at Waterville, Ohio.

There are situations, particularly when the data set is small (for example, 100 or 200 observations) or the curves are being extended to a range where there are very few observations, that the curves produced acquire a "sawtooth" appearance, rather than the smooth curves in figures 42 and 43. The following command produces such an example:

```
plotConcQSmooth(eList, "2010-04-01","2010-08-01","2010-12-01", 1, 300, legend-
Left=10, legendTop=0.19, logScale = TRUE)
```

The result is depicted in figure 44.

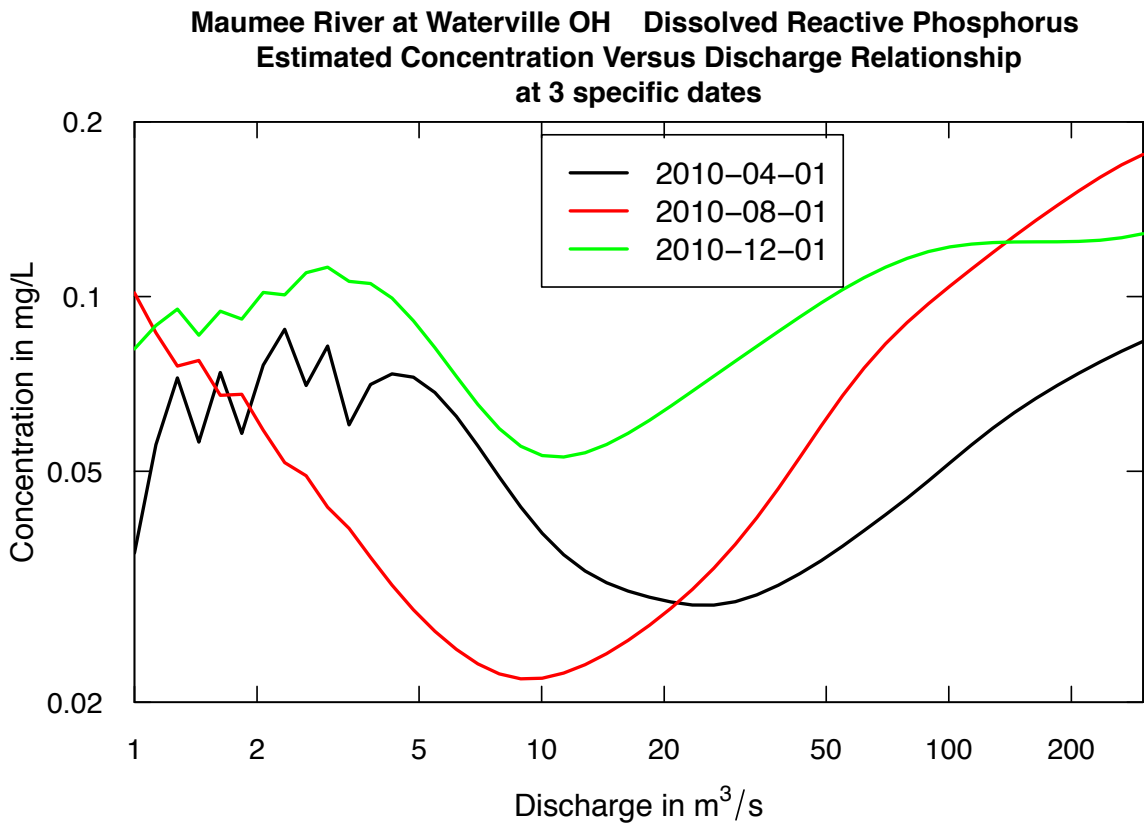


Figure 44. Plot produced by the `plotConcQSmooth` function for the relation between concentration of dissolved reactive phosphorus and discharge centered on three dates (April 1, August 1, and December 1) during 2010 for the Maumee River at Waterville, Ohio, but shown with a discharge scale extended to excessively low values.

The explanation of the sawtooth pattern at the lowest discharge values is that the estimates represented by these curves, particularly those less than discharge values around 8 m^3/s, are at such extreme edges of the full sample data set that the WRTDS algorithm has to widen the windows (one or more times) to obtain the required 100 observations with nonzero weights. This incremental widening creates a series of oscillations in the curves that are simply artifacts of the widening algorithm. Typically, when these kinds of oscillations happen, it is because the range of discharge values considered goes too far towards the extremes of the data set. In this case, for example, the lowest observed discharge in the sample data set is 1.56 m^3/s, so the curves are extending into a range where there are very few observations. Resetting the `qLow` argument to 30 resolves this problem for this data set.

In any given application of the WRTDS model there will be portions of the model domain where the combination of discharge and time of year are such that the available data that are close to those values are very limited, and the kind of oscillatory behavior shown here does influence the values that are stored in the `surfaces` matrix. This oscillatory behavior has two consequences:

1. Plotted versions of the surfaces can be rather unreliable at the most extreme low or high discharges. Therefore, applications of `plotContours` and `plotDiffContours` should not encompass the full extent of the domain over which they were calculated, but should be compressed to the more common values; for example, between the 5-percent and 95-percent values on the flow-duration curve. If possible, avoid the first and last years of the data set.

2. They will influence the computation of estimated concentrations and fluxes. Although these oscillations can have a moderately large influence on individual daily estimates, when they are aggregated into monthly or annual mean values of concentration or flux, they should balance out and have very little influence on these aggregated results. Sensitivity tests have shown that annual results are rather insensitive to reasonable modifications of the smoothing parameters, even though they can cause a noticeable change in these `plotConcQSmooth` graphs.

When data sets are rather small (for example, less than 100 observations), the two arguments `minNumObs` and `minNumUncen` may need to be reduced from their default values of 100 and 50, respectively. Use of alternate values of these two arguments in `plotConcQSmooth` graphs can indicate the degree of sensitivity of results to these changes. All of these adjustments of the smoothing parameters in this function produce results that are not derived from the smooth estimated concentration surfaces stored in the object `surfaces`. These `plotConcQSmooth` curves are instead calculated by performing the weighted regressions at each of 48 discharge values for each curve plotted, and they use the smoothing parameters specified in the given command. Changing these parameters in `plotConcQSmooth` does not result in a change in the `surfaces` object and, hence, does not result in any changes in the calculated monthly or annual results. If these plots suggest the need to change the smoothing parameters to be used to compute the WRTDS results, then the function `modelEstimation` must be rerun with the desired smoothing parameters (the arguments `windowQ, windowY, windowS, minNumObs, minNumUncen,` and `edgeAdjust`). When `modelEstimation` is rerun with new parameters, these parameter values are all stored in the `INFO` data frame, and new versions of `Daily, Sample, surfaces`, and `AnnualResults` replace the versions that had been computed the last time `modelEstimation` was run.

plotConcTimeSmooth

As its name suggests, the `plotConcTimeSmooth` function operates very similarly to `plotConcQSmooth`, but it "slices" the contour plot in the horizontal direction. However, if one were to take a slice through the contour plots at a given discharge across the full time range, the resulting curve would show a strong seasonal pattern. To emphasize the long-term trend pattern, this seasonal oscillation is eliminated by the selection of a particular day of the year to be plotted. A way to think about the construction of this graphic is this: If one plotted a horizontal line on the contour plot for estimated concentration at a selected discharge and then plotted a set of vertical lines on the same day of each year in the record, the `plotConcTimeSmooth` graph would capture the value of estimated concentration at every intersection of the horizontal and vertical lines. This can be done for as many as three different discharge values in one graph. Caution should always be used when interpreting these curves near the starting and ending dates of the data set. Smoothing algorithms, such as the one used here, will always be less reliable at the extreme ends of the record than they are in the middle. Many of the arguments that are used in `plotConcQSmooth` are used in this function, but for the sake of completeness, the information about all of the arguments is presented here as they were in table 11.

Table 12. Arguments for the function `plotConcTimeSmooth`.

Argument	Definition	Default
`eList`	Specifies the named list that contains the `INFO` and `Sample` data frames that will be used by this function.	No default, name of list must be stated.
`q1`	The discharge, in units specified by `qUnit`, that specifies the discharge for the first curve on the figure.	No default is established, the user must set this value.
`q2`	The discharge, in units specified by `qUnit`, that specifies the discharge for the second curve on the figure. If a second curve is not to be plotted, then `q2` should be set to NA.	No default is established, the user must set this value.
`q3`	The discharge, in units specified by `qUnit`, that specifies the discharge for the third curve on the figure. If a third curve is not to be plotted, then `q3` should be set to NA.	No default is established, the user must set this value.
`centerDate`	This argument sets the month and day of the year for which all of the curves are to be computed. The argument must be specified in the form "mm-dd," thus if the curve is to be computed for May 15 of every year, `centerDate = "05-15"` (it must be in quotes).	No default is established, the user must set this value
`yearStart`	This is the starting year for the graph. The first plotted value will be on the first `centerDate` after the time specified by `yearStart`.	No default is established, the user must set this value.
`yearEnd`	This is the ending year for the graph. The last plotted value will be on the `centerDate` prior to the time specified by `yearEnd`.	No default is established, the user must set this value.

Table 12. Arguments for the function `plotConcTimeSmooth`.—Continued

Argument	Definition	Default
`qUnit`	Determines the units to be used for discharge. See `printqUnitCheatSheet()` for a list of discharge unit codes.	`qUnit = 2`. Discharge is in m^3/s.
`legendLeft`	The location of the left edge of the legend, expressed as time in years.	`legendLeft` = 0 (this allows the function to set `legendLeft` automatically).
`legendTop`	The location of the top edge of the legend, expressed in concentration units (mg/L).	`legendTop` = 0 (this allows the function to set `legendTop` automatically).
`printLegend`	A logical variable; if `TRUE`, legend is included; if `FALSE`, legend is not included	printLegend = TRUE
`concMax`	Upper bound on the vertical axis for plot. This can be useful when multiple plots are being compared.	`concMax = NA`, which results in the program selecting the upper bound based on the curves being shown.
`concMin`	Lower bound on the vertical axis for the plot. This will be ignored if the vertical scale is arithmetic (`logScale = FALSE`), in which case the lower bound is always 0 mg/L. This can be useful when multiple plots are being compared.	`concMin = NA`, which results in the program selecting the lower bound based on the curves being shown (if `logScale = TRUE`), and results in a lower bound of 0 (if `logScale = FALSE`).
`bw`	A logical variable. If bw = FALSE, curves are plotted in color. If bw = TRUE, curves are plotted in black and white.	bw = FALSE
`printTitle`	A logical variable. If `printTitle = TRUE`, title is printed. If `printTitle = FALSE`, title is not printed.	printTitle = TRUE
`printValues`	A logical variable. If `printValues = TRUE`, in addition to plotting the graphic, the function prints the values plotted to the console and creates a data frame with these values (see discussion below)	printValues = FALSE
`windowY`	Half-window width for time, in years, in the WRTDS smoothing method. (See notes on selection of smoothing parameters below)	windowY = 7
`windowQ`	Half-window width for discharge, in natural log units, for the WRTDS smoothing method. (See notes on selection of smoothing parameters below)	windowQ = 2
`windowS`	Half-window width for seasons, in years, for the WRTDS smoothing method. (See notes on selection of smoothing parameters below)	windowS = 0.5
`minNumObs`	Minimum number of observations required to run a specific weighted regression. (See notes on selection of smoothing parameters below)	minNumObs = 100
`minNumUncen`	Minimum number of uncensored observations required to run a specific weighted regression. (See notes on selection of smoothing parameters below)	minNumUncen = 50
`logScale`	A logical variable. If `logScale = TRUE`, vertical scale is a log scale. If `logScale = FALSE`, vertical scale is arithmetic and starts at 0 mg/L.	logScale = FALSE
`edgeAdjust`	A logical variable. If `TRUE` the half window width for time weighting is increased when the estimation time is less than `windowY` years from the beginning or end of the estimation period. It is adjusted so that the full width of the window is equal to 2*`windowY`. If FALSE, the window width is not adjusted near the beginning or end of the data set. If FALSE, the optimal `windowY` value may be closer to 10 rather than 7 years.	edgeAdjust = TRUE

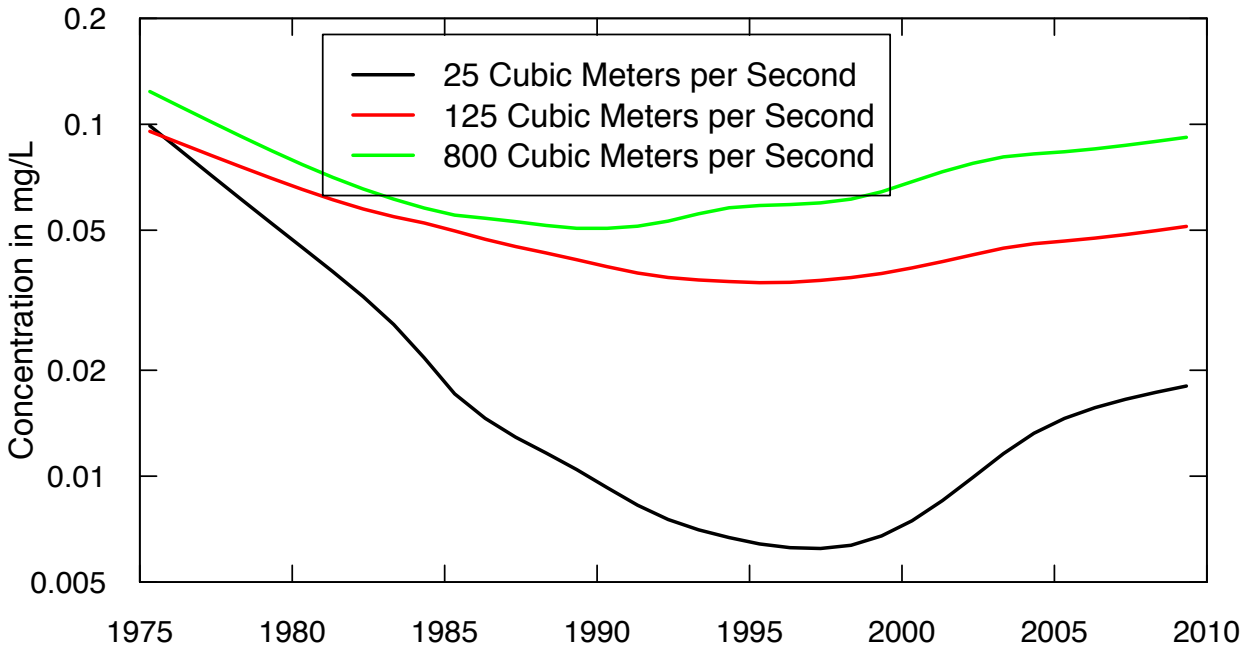


Figure 45. Concentration versus time plots produced by the `plotConcTimeSmooth` function for the relation between concentration of dissolved reactive phosphorus at three different discharge values, centered on May1 of each year, for the Maumee River at Waterville, Ohio.

Use of the following command:

```
plotConcTimeSmooth(eList, q1=25, q2=125, q3=800, centerDate="05-01",
yearStart=1975, yearEnd=2010, logScale=TRUE, legendLeft=1981, legendTop=0.18,
printTitle=FALSE)
```

results in the graphic shown in figure 45.

The plot reveals that for early May conditions, concentrations of DRP declined very markedly for low a discharge value such as 25 m^3/s, which is near the 5th percentile point on the seasonal flow-duration curve, from 1975 to 1997, with a total decline of about 94 percent followed by an increase of about 190 percent from 1997 to 2009. The decrease over the entire time period is about 82 percent. At a moderate discharge of 125 m^3/s, which is approximately the median discharge for this time of year, the initial decrease from 1975 to 1995 was 63 percent. Since that time, the mean concentration at this time of year and discharge has risen by about 40 percent. However, for high discharge values of 800 m^3/s, which is the 95th percentile of the seasonal flow-duration curve, the decline from 1975 through 1989 was about 60 percent. But, from 1989 through 2009 there has been an increase of about 80 percent in the mean concentration for this discharge and time of year.

This example is illustrative of just how different the patterns of trend can be across the range of streamflow conditions. The widely used LOADEST model assumes that trends in log concentration are quadratic in shape (these are certainly not quadratic) and that they assume the same shape regardless of discharge (again, very far from true). The Maumee River is an example where there has been great success in reducing point-source contributions, but the nonpoint sources, which are most important at high discharge, have increased over time. Graphics produced by `plotConcTimeSmooth` can help elucidate the time history of changes in water quality at various discharge levels and can help the user interpret the record in terms of understanding the importance of various types of pollutant sources and evaluating the effectiveness of different control strategies because those strategies are likely to have different "signatures" in terms of the time of year and discharge conditions at which they can be expected to me most effective.

Editing Data Sets

The R programming language is a powerful environment for data manipulation, and many helpful resources are available in books and Web sites. A few simple examples will be given here, but these examples are in no way a comprehensive introduction to R data manipulation.

The structure of many of the EGRET objects as described throughout this report are data frames. To introduce some basic concepts of data frames, let us assume we have a data frame called `DF`. In data frame `DF`, there are three columns: `Date`, `Value`, and `Qualifier`. An individual entry in the data frame can be called in several ways. One way is to use a single set of brackets, inside of which row and column are defined. For example, `DF[1,2]` outputs the first row, second column. Therefore, if we simply wanted to view the data for that cell, we could type:

```
DF[1,2]
```

If we wanted to adjust the value in that cell, we could assign a new value:

```
DF[1,2] <- 5
```

With this convention, we could also get all rows of the second column:

```
DF[,2]
```

Alternatively, all columns of a particular row (the first row in this example):

```
DF[1,]
```

Another way to view the same data is to use the column name. `DF$Value` would return a vector with all the values of the column 'Value'. To view the same data as above (`DF[1,2]`), we could also use the command:

```
DF$Value[1]
```

If we want to adjust the value to that cell, we could assign a new value:

```
DF$Value[1] <- 6
```

R also lets you put logical statements within the square brackets. So, if you wanted a subset of data that was greater than 10, you could use the command:

```
DF[DF$Value > 10,]
```

If we want to replace values that are reported as less than or equal to 10 to have the `Qualifier` "<" and a `Value` of `10`, then the following commands would work:

```
DF$Value[DF$Value <= 10] <- 10
DF$Qualifier[DF$Value == 10] <- "<"
```

With this basic knowledge of R data frame structure, we can work through a few examples that might appear in an EGRET analysis.

Working with eList

The named list, called `eList`, is considered an EGRET object. It contains up to four objects, always in the same order (`INFO`, `Daily`, `Sample`, `surfaces`), although some of them can be missing. Creating `eList` from the individual objects can be done with the following command:

```
eList <- as.egret(INFO, Daily, Sample, surfaces)
```

but depending on the situation, it might be created using `NA` where the name of the object belongs.

If we want to examine one of the objects (lets say `Sample`) then we could do so with the command `summary(eList$Sample)` or we can print out a very specific part of the object. For example, the command, `eList$Sample$ConcAve[1:10]` would print the first ten values in the ConcAve column of `Sample`.

However, if we want to edit any of these objects once they have been joined together in `eList`, a copy of that object should be made. For example, if we wanted to look at `Sample` after it has been processed in `modelEstimation`, we can give the command.

```
Sample <- eList$Sample
```

Then `Sample` can be examined by giving the command `summary(Sample)` or a more detailed command such as `Sample$ConcAve[1:10]` (which would produce the same result as the `eList$Sample$ConcAve[1:10]` command).

Once the copy exists, the editing methods described below can be applied to that object. Then, once all of the editing is complete, the `eList` can be reassembled. For example, if we have edited only `Sample` we can remake `eList` with the command `eList <- as.egret(eList$INFO, eList$Daily, Sample, NA)`. The user should allow the function `modelEstimation` to create the new `surfaces` object and place it in `eList`. If `INFO`, `Daily`, and `Sample` have all been edited, then `eList` should be reassembled with the command `eList <- as.egret(INFO, Daily, Sample, NA)`.

Deleting Values

If we had some reason to believe that all the samples with concentration values above 100 were suspect, we could delete those rows from the `Sample` data frame:

```
Sample <- Sample[Sample$ConcAve < 100,]
```

There is also function built into R, `subset` that would accomplish the same goal:

```
Sample <- subset(Sample,ConcAve < 100)
```

Once the Sample data frame has been modified, the eList object needs to be updated:

```
eList$Sample <- Sample
```

Shortening the Period of Record

In the `Daily` and `Sample` data frames, there is a column called `Date`. This column is in an R date structure, which means it contains more information than just the character string. We can compare the dates in the `Date` columns with other dates. The simplest way to create an R date object is to use a string in the format "YYYY-MM-DD", enclosed in `as.Date()`. If we had a `Sample` data frame and we wanted to remove all samples that were collected before Jan. 1, 1980, we could use the following command:

```
smallSample <- Sample[as.Date("1980-01-01") < Sample$Date,]
```

or:

```
smallSample <- subset(Sample,as.Date("1980-01-01") < Date)
eList$Sample <- smallSample
```

Creating Interval Concentrations

Figure 13, which shows the full set of concentration data for the Choptank River at Greensboro, Maryland, clearly shows that the rounding convention for reporting the data changed around 1994. For the earlier part of the record, values measured at 1.0 mg/L or greater were rounded to the nearest 0.1 mg/L, and lower values, or those that came later in the record, were reported to a higher degree of precision (generally the nearest 0.01 mg/L). It might be of interest to consider a way of expressing the

values from the earlier period that reflects the true degree of precision. This could be done by considering those rounded values to be interval censored data, with the interval covering a range from 0.05 mg/L below the reported value to 0.05 mg/L above the reported value (so a value reported as 1.2 mg/L becomes an interval of 1.15 to 1.25 mg/L). The following steps could accomplish this kind of change:

```
Sample$ConcLow <- ifelse(as.Date(Sample$Date) < "1994-10-01" & Sample$ConcLow >=
1.0, Sample$ConcLow-0.05, Sample$ConcLow)
```

The `ifelse` function is a way of applying some logical test to a vector of values, and in the vector that is returned, it is set to one value if the answer is TRUE and another value is the answer is FALSE. In this case it is being used to reset the values in `Sample$ConcLow`. The test checks two conditions: the first is that the date is before 1994-10-01, and the second is that the existing value of `Sample$ConcLow` is greater than or equal to 1. If these two conditions hold, then we want to apply interval censoring, and if one or both of the conditions do not hold, then we do not want to apply interval censoring. In the TRUE case, we set the new value to be 0.05 below the reported value of `Sample$ConcLow`. In the FALSE case, we leave the `Sample$ConcLow` value at its existing value.

The next step is to make the same test and adjustment to `Sample$ConcHigh`. The command is very similar:

```
Sample$ConcHigh <- ifelse(as.Date(Sample$Date) < "1994-10-01" & Sample$ConcHigh >=
1.0, Sample$ConcHigh+0.05, Sample$ConcHigh)
```

Now that changes have been made in some of the `Sample$ConcLow` and `Sample$ConcHigh` values, the `Sample$ConcAve` and `Sample$Uncen` values need to be recomputed and stored in the revised data frame. The function `fixSampleFrame` will accomplish this. After completing our editing of the `Sample` data frame we need to put the new version back into `eList`. We can do it with the command:

```
eList$Sample <- Sample
```

Then the following command will make the needed adjustments of `Sample$ConcAve` and `Sample$Uncen`:

```
eList <- fixSampleFrame(eList)
```

These are just a few examples of how R functions can be used to make certain kinds of edits to the data. It is often wise to keep a copy of the data set as it was before any editing was done, and this is easily accomplished in R by making an unedited copy of the specific data frame to keep. For example, if we want to keep a copy of Sample in its original form, we could create a new data frame called `keepSample` by giving the command:

```
keepSample <- Sample
```

Then we would proceed to do our editing on the version called `Sample`. If, later on, we wished to revert to using the original data frame version of the data set, this could be accomplished with the command:

```
Sample <- keepSample
eList$Sample <- keepSample
```

Working with Multiple Versions of Data Frames

Throughout this User Guide up to this point, the named list has always been `eList`. However, it can actually take on any name the user selects. If the user wants to consider a variety of alternatives to their analysis, this can easily be done by using a different name for each alternative. The alternatives might be: different ways of handling “less than” values (we may be uncertain about what the true reporting limit was), eliminating or revising one or more suspect data values, eliminating a portion of the record, thinning out the record in some period when sampling was much more intensive than normal, using interval censoring to represent some values, or modifying some of the WRTDS parameters such as window widths or minimum numbers of samples to be used in each regression.

Let’s look at a real example. In the Choptank River example, we retrieved all of the samples back to 1979-09-01. Let us say that we are looking at this as a part of a larger study, and the other records in the study did not start until water year 1985, and we

want to consider the idea that we might want to leave out the Choptank data prior to 1985 and see if it substantially influences our results. First, a comment about this idea: It is certain that there will be some influence on the later results, because the model for 1985 and beyond is influenced by data from before 1985 (because of the windows). The question is, how large is that influence? A second comment is that there is probably no right or wrong answer as to how to proceed with such a study. The case for using the earlier data is that one should always work with the greatest amount of information available. The case against it is that if we used the earlier data, then this one site would show behaviors around 1985 that are influenced by events such as regional climatic events and (or) regional changes in land-use or cultivation practices that preceded 1985 but were excluded from the other data sets, because such data were not available. In any event, we may wish to make this long-record/short-record comparison. Let us assume here that we have already run the retrievals, fit the model, and created the `Sample`, `Daily`, and `INFO` data frames and have the results in the form of `surfaces`, all of which are contained in `eList`.

In this situation, we would pull out `INFO`, `Daily`, and `Sample` from `eList`, as follows.

```
INFO <- eList$INFO
Daily <- eList$Daily
Sample <- eList$Sample
```

We may want to change `INFO` by simply adding some text to explain the nature of the change:

```
INFO$note <- "data edited to exclude data prior to WY 1985"
```

We want to modify `Daily` and `Sample` to eliminate the earlier data. We can do that with these two commands.

```
Daily <- subset(Daily, Date > "1984-09-30")
Sample <- subset(Sample, Date > "1984-09-30")
```

We may want to check to see if things look correct by doing `summary(Daily)` and `summary(Sample)`. Then we can put these three data frames into a new list called, for example, `eListLater`. The command would be:

```
eListLater <- as.egret(INFO, Daily, Sample, NA)
```

At this point, we can proceed with all of the usual steps for exploring the data and looking at results. For example we might do:

```
multiPlotDataOverview(eListLater)
eListLater <- modelEstimation(eListLater)
plotConcHist(eListLater)
tableResults(eListLater)
```

And we might want to construct graphs comparing, for example, the annual flow-normalized concentrations from the original analysis and from this alternative analysis. We can do this by creating data frames of each set of results:

```
annualResults <- setupYears(eList)
annualResultsLater <- setupYears(eListLater)
```

We can then produce graphs, the first one being `annualResults$DecYear` versus `annualResults$FNConc` and the second one being `annualResultsLater$DecYear` versus `annualResultsLater$FNConc`.

By saving `INFO`, `Daily`, `Sample`, and `surfaces` together in a single list we avoid any possible confusion of what version of the results go with what version of the data and we can return to these two lists, `eList` and `eListLater`, to make whatever comparisons we wish to make.

If we wanted to make a comparison using different WRTDS parameters (as opposed to comparisons with differences in the data), we can use this approach. Say in our original analysis (stored in `eList`) we used all the default parameters in `modelEstimation`, but now we want to consider `windowY = 10`, `windowQ = 1.5`, and `minNumObs = 60`. We could use this command:

```
eListAlt <- modelEstimation(eList, windowY = 10, windowQ = 1.5, minNumObs = 60)
```

When returned, all four of the objects in `eListAlt` will be different from the versions in `eList`. `eListAlt$INFO` will contain information about these WRTDS parameters: `eListAlt$Daily` will have the daily estimates from the alternative model, `eListAlt$Sample` will have the cross-validation estimates from the alternative model, and `eListAlt$surfaces` will be the new surfaces estimated. We can explore the results with all of the usual functions (such as `plotConcHist` or `plotContours`) simply by putting in `eListAlt` as the first argument in the function. All of these results can be saved for future examination by using the `saveResults` function.

Batch Processing in EGRET

When the user runs a large number of analyses, the EGRET code can be run in batch mode by creating a text file that contains the sequence of commands that are to be run, and this file can then be run by using the `source` function in R. The process of creating the initial workspace (the data frames `Daily`, `Sample`, and `INFO`) is best done interactively, although they can be created in batch. The reasons for doing these steps interactively are 1) these steps typically take very little computer time, and 2) the interactive approach is more conducive to identifying data problems such as highly unusual data values or record lengths that are different from what may be needed for the analysis being planned. Running the process through the step of `multiPlotDataOverview` gives the analyst several opportunities to spot problems with the data set that need to be explored and corrected.

Let's assume that we have created two workspaces, `CHOP.TN.RData` for Choptank River total nitrogen and `SUSQ.TN.RData` for Susquehanna River total nitrogen. These workspaces each contain the `Daily`, `Sample`, and `INFO` data frames, and they have been placed into eList using the command `eList <- mergeReport(INFO, Daily, Sample)` and everything is ready for running model estimation and looking at results. Let's assume that we have a directory where these workspaces are saved and the object `savePath` has been defined with a command like:

```
savePath<-"/Users/rhirsch/Desktop/examples/"
```

Here is what the file of commands might look like:

```
library(EGRET)
fileName<-paste(savePath,"CHOP.TN.RData",sep="")
load(fileName)
eList <- modelEstimation(eList)
tableChange(eList, fluxUnit=5,yearPoints=c(1980,1990,2000,2010))
tableResults(eList, qUnit=1,fluxUnit=5)
saveResults(savePath, eList)

fileName<-paste(savePath,"SUSQ.TN.RData",sep="")
load(fileName)
eList <- modelEstimation(eList)
tableChange(eList, fluxUnit=6,yearPoints=c(1980,1990,2000,2010))
tableResults(eList, qUnit=3,fluxUnit=6)
saveResults(savePath, eList)
```

We can also achieve the same effect by using a loop, which is especially useful if we are running a large number of EGRET analyses:

```
library(EGRET)
savePath<-"/Users/rhirsch/Desktop/examples/"
filesToOpen <- c("CHOP.TN.RData","SUSQ.TN.RData")
flux <- c(5,6)
q <- c(1,3)


for (i in 1:length(filesToOpen)){
  fileName<-paste(savePath, filesToOpen[i],sep="")
  load(fileName)
  eList <- modelEstimation(eList)
  tableChange(eList, fluxUnit=flux[i],yearPoints=c(1980,1990,2000,2010))
```

```
 tableResults(eList, qUnit=q[i],fluxUnit=flux[i])
 saveResults(savePath, eList)
}
```

This code will load the workspace, run the WRTDS analysis, provide tabular output (to the console), and then resave the workspace with the new eList object. Note that some of the options in the table commands are different because the rivers are of very different sizes, so we might want to see flux in tons per year for the Choptank and thousands of tons per year for the Susquehanna. Later on, the workspaces that we saved can be reloaded and a variety of plots can be made. Note that most of the batch commands are just a repetition of steps, and we need to enter only a few differences to make up a long file of many analyses. Then, if we name this file "exampleCommands.txt", all we need to do once we start up R is to give the one command:

```
source("<full pathname goes here>/exampleCommands.txt")
```

Appendix 3 shows an example workflow that produces PDFs of all the graphics and text files for all the tables in one folder.

References Cited

Baker, D.B., and Richards, R.P., 2002, Phosphorus budgets and riverine phosphorus export in Northwestern Ohio Watersheds: Journal of Environmental Quality, v. 31, p. 96–108.

Cleveland, W.S., 1979, Robust locally weighted regression and smoothing scatterplots: Journal of the American Statistical Association, v. 74, no. 368, p. 829–836.

Cleveland, W.S., and Devlin, S.J., 1988, Locally weighted regression—An approach to regression analysis by local fitting: Journal of the American Statistical Association, v. 83, no. 403, p. 596–610.

Cohn, T.A., Caulder, D.L., Gilroy, E.J., Zynjuk, L.D., and Summers, R.M., 1992, The validity of a simple statistical model for estimating fluvial constituent loads: An empirical study involving nutrient loads entering Chesapeake Bay: Water Resources Research, v. 28, no. 9, p. 2353–2363.

Debrewer,, L., Ator, S., and Denver, J., 2008, Temporal trends in ntrate and selected pesticides in Mid-Atlantic ground water: Journal of Environmental Quality, v. 37, no. 5,Suppl., p. S–296–S–308.

Garrett, J.D., 2012, Concentrations, loads, and yields of select constituents from major tributaries of the Mississippi and Missouri rivers in Iowa, water years 2004–2008: U.S. Geological Survey Scientific Investigations Report 2012–5240, p. 61, *http://pubs.usgs.gov/sir/2012/5240*.

Gordon, N.D., McMahon, T.A., and Finlayson, B.L., 1991, Stream hydrology: an introduction for ecologists: New York, John Wiley and Sons Ltd., 429 p.

Helsel, D.R., and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water Resources Investigations, book 4, chap. A3, 522 p., *http://pubs.usgs.gov/twri/twri4a3/*.

Hirsch, R.M., 2012, Flux of nitrogen, phosphorus, and suspended sediment from the Susquehanna River Basin to the Chesapeake Bay during Tropical Storm Lee, September 2011, as an indicator of the effects of reservoir sedimentation on water quality: U.S. Geological Survey Scientific Investigations Report 2012–5185, 17 p., *http://pubs.usgs.gov/sir/2012/5185/*.

Hirsch, R.M., 2014, Large biases in regression-based constituent flux estimates: Causes and diagnostic tools: Journal of the American Water Resources Association, v. 50, no. 6, p. 1401–1424, *http://onlinelibrary.wiley.com/doi/10.1111/jawr.12195/full*.

Hirsch, R.M., and Fisher, G.T., 2014, Past, present, and future of water data delivery from the U.S. Geological Survey: Journal of Contemporary Water Research and Education, Universities Council on Water Resources, no. 153, p. 4–15, *http://www.ucowr.org/issue-153-water-data/past-present-and-future-of-water-data-delivery-from-the-us-geological-survey*.

Hirsch, R.M., Moyer, D.L., and Archfield, S.A., 2010, Weighted Regressions on Time, Discharge, and Season (WRTDS), with an application to Chesapeake Bay River inputs: Journal of the American Water Resources Association, v. 46, no. 5, p. 857–880, *http://onlinelibrary.wiley.com/doi/10.1111/j.1752-1688.2010.00482.x/full.*

Hirsch, R.M., Slack, J.R., and Smith, R.A., 1982, Techniques of trend analysis for monthly water quality data: Water Resources Research, v. 18, no. 1, p. 107–121.

International Joint Commission, 2014, A balanced diet for Lake Erie: reducing phosphorus loadings and harmful algal blooms: Report of the Lake Erie Ecosystem Priority, Washington, D.C., and Ottawa, Ont., 100 p., *http://www.ijc.org/files/publications/2014%20IJC%20LEEP%20REPORT.pdf*

Michalak, A.M., Anderson, E.J., Beletsky, D., Boland, S., Bosch, N.S., Bridgeman, T.B., Chaffin, J.D., Cho, K., Confesor, R., and Daloğlu, I., 2013, Record-setting algal bloom in Lake Erie caused by agricultural and meteorological trends consistent with expected future conditions: Proceedings of the National Academy of Sciences, v. 110, no. 16, p. 6448–6452, *http://www.pnas.org/content/110/16/6448.full*.

Montgomery, D.C., Peck, E.A., and Vining, G.G., 2012, Introduction to linear regression analysis (5th ed.): Hoboken, N.J., John Wiley and Sons, 672 p.

Moyer, D., Hirsch, R., and Hyer, K., 2012, Comparison of two regression-based approaches for determining nutrient and sediment fluxes and trends in the Chesapeake Bay watershed: U.S. Geological Survey Scientific Investigations Report 2012–5244, 118 pp., *http://pubs.usgs.gov/sir/2012/5244/*.

Murphy, J.C., Hirsch, R.M., and Sprague, L.A., 2014, Antecedent flow conditions and nitrate concentrations in the Mississippi River basin: Hydrology and Earth System Sciences, v. 18, no. 3, p. 967–979, *http://www.hydrol-earth-syst-sci.net/18/967/2014/hess-18-967-2014.pdf*.

Richards, R.P., Alameddine, I., David Allan, J., Baker, D.B., Bosch, N.S., Confesor, R., DePinto, J.V., Dolan, D.M., Reutter, J.M., and Scavia, D., 2012, "Nutrient inputs to the Laurentian Great Lakes by source and watershed estimated using SPARROW watershed models" by Dale M. Robertson and David A. Saad: JAWRA Journal of the American Water Resources Association, no. 49, p. 715–724, *http://onlinelibrary.wiley.com/doi/10.1111/jawr.12006/abstract*.

Rice, K.C., and Hirsch, R.M., 2012, Spatial and temporal trends in runoff at long-term streamgages within and near the Chesapeake Bay Watershed: U.S. Geological Survey Scientific Investigations Report 2012–5151, 56 p., *http://pubs.usgs.gov/sir/2012/5151/*.

Richards, R.P., and Baker, D.B., 2002, Trends in water quality in LEASEQ rivers and streams (Northwestern Ohio), 1975–1995: Journal of Environmental Quality, v. 31, no. 1, p. 90–96.

Riggs, H.C., 1982, Low-flow investigations: U.S. Geological Survey Techniques of Water-Resources Investigations Report, book 4, chap. B1, 23 p., *http://pubs.usgs.gov/twri/twri4b1/*.

Runkel, R., Crawford, C., and Cohn, T.A., 2004, Load Estimator (LOADEST): A FORTRAN Program for estimating constituent loads in streams and rivers: U.S. Geological Survey Techniques and Methods, book 4, chap. A5, 69 p., *http://pubs.usgs.gov/tm/2005/tm4A5*/.

Scott, J., Gellenbeck, D., Young, D., and Booth, N., 2008, U.S. federal water quality web service collaboration: Eos, v. 86, issue 52, p. 543–544, at *http://onlinelibrary.wiley.com/doi/10.1029/2008EO520003/abstract*.

Sprague, L.A., Hirsch, R.M., and Aulenbach, B.T., 2011, Nitrate in the Mississippi River and its tributaries, 1980 to 2008: Are we making progress?: Environmental Science & Technology, v. 45, no. 17, p. 7209–7216, *http://pubs.acs.org/doi/pdf/10.1021/es201221s*.

Sprague, L.A., Langland, M.J, Yochum, S.E., Edwards, R.E., Blomquist, J.D., Phillips, S.W., Shenk, G.W., and Preston, S.D., 2000, Factors affecting nutrient trends in major rivers of the Chesapeake Bay watershed: U.S. Geological Survey Water Resources Investigations Report 00–4218, 109 p., *http://va.water.usgs.gov/online_pubs/WRIR/00-4218.htm*.

Stenback, G.A., Crumpton, W.G., Schilling, K.E., and Helmers, M.J., 2011, Rating curve estimation of nutrient loads in Iowa rivers: Journal of Hydrology, v. 396, no. 1, p. 158–169, *http://dx.doi.org/doi:10.1016/j.jhydrol.2010.11.006*.

Tobin, J., 1958, Estimation of relationships for limited dependent variables: Econometrica: journal of the Econometric Society, p. 24–36.

Tukey, J.W., 1977, Exploratory data analysis: Reading, Mass., Addison-Wesly, 689 p.

Appendixes 1–4

Appendix 1. dataRetrieval Vignette (separate PDF)

Appendix 2. EGRET Vignette (separate PDF)

Appendix 3. Batch Workflows

Appendix 4. Sample Workflow

Appendix 3. Batch Workflows

These scripts show how a plan for batch processing might be structured. They also provide a reference of common basic workflows that users might follow interactively.

```
library(dataRetrieval)
library(EGRET)

# list of sites
sites <- c("01491000", "11264500")
savePath <- "D:/LADData/RCode/Example1/"
fileNames <- rep(NA,length(sites))

for (i in 1:length(sites)){
  ############################
  # Gather discharge data:

  startDate <- "" #Gets earliest date
  endDate <- "2011-09-30"
  Daily <- readNWISDaily(sites[i],"00060",startDate,endDate)

  # Gather sample data:
  parameter_cd<-"00631" #5 digit USGS code
  Sample <- readNWISSample(sites[i],parameter_cd,startDate,endDate)

  # Gather site and parameter information:
  INFO<- readNWISInfo(sites[i],parameter_cd,interactive=FALSE)
  INFO$staAbbrev <- INFO$station_nm

  # Merge discharge with sample data and create eList:
  eList <- mergeReport(INFO, Daily, Sample, NA, interactive=FALSE)
  ############################

  fileNames[i] <- paste(savePath, INFO$staAbbrev, ".",
                        INFO$constitAbbrev,".RData", sep = "")
  saveResults(savePath, eList)
}

for (i in fileNames){

  load(i)

  pdf(paste(savePath,INFO$staAbbrev,
            "_FlowHistoryPlots.pdf",sep=""))
  ############################
  # Check flow history data:
  plotFlowSingle(eList, istat=7,qUnit="thousandCfs")
  plotSDLogQ(eList)
  plotQTimeDaily(eList, qLower=1,qUnit=3)
  plotFour(eList, qUnit=3)
  plotFourStats(eList, qUnit=3)
  ############################
  dev.off()
```

```
pdf(paste(savePath,INFO$staAbbrev,
          "_ConcHistoryPlots.pdf",sep=""))
############################
# Check sample data:
boxConcMonth(eList)
boxQTwice(eList)
plotConcTime(eList)
plotConcQ(eList)
multiPlotDataOverview(eList)
############################
dev.off()
############################
# Run WRTDS model:
eList <- modelEstimation(eList)
############################

pdf(paste(savePath,INFO$staAbbrev,
          "_WRTDS_Plots.pdf",sep=""))
  ############################
  #Check model results:

  plotConcTimeDaily(eList)
  plotFluxTimeDaily(eList)
  plotConcPred(eList)
  plotFluxPred(eList)
  plotResidPred(eList)
  plotResidQ(eList)
  plotResidTime(eList)
  boxResidMonth(eList)
  boxConcThree(eList)

  #Explore trend results:
  plotConcHist(eList)
  plotFluxHist(eList)

  # Multi-line plots:
  date1 <- "2000-09-01"
  date2 <- "2005-09-01"
  date3 <- "2009-09-01"
  qBottom<-100
  qTop<-5000
  plotConcQSmooth(date1, date2, date3, qBottom, qTop,
                  concMax=2,qUnit=1)
  q1 <- 10
  q2 <- 25
  q3 <- 75
  centerDate <- "07-01"
  yearEnd <- 2010
  yearStart <- 2000
  plotConcTimeSmooth(q1, q2, q3, centerDate, yearStart, yearEnd)

  # Multi-plots:
  fluxBiasMulti(eList)
```

```
    #Contour plots:
    clevel<-seq(0,2,0.5)
    maxDiff<-0.8

    plotContours(eList, yearStart,yearEnd,qBottom,qTop,
contourLevels = clevel,qUnit=1)
    plotDiffContours(eList, yearStart,yearEnd,
                     qBottom,qTop,maxDiff,qUnit=1)
  dev.off()

  concChange <- tableChangeSingle(eList, returnDataFrame=TRUE,flux=FALSE)
  write.csv(concChange,file=paste(savePath,INFO$staAbbrev,
                                  "_concChange.csv",sep=""))
  fluxChange <- tableChangeSingle(returnDataFrame=TRUE,flux=TRUE)
  write.csv(fluxChange,file=paste(savePath,INFO$staAbbrev,
                                  "_fluxChange.csv",sep=""))

  sink(paste(savePath,INFO$staAbbrev,"_tableFlowChange.txt",sep=""))
  tableFlowChange(eList, istat=1)
  sink()

  saveResults(savePath, eList)

}
```

Appendix 4. Sample Workflow

These workflows illustrate a simplified workflow used in interactive processing. They can serve as a handy reference to remind the analyst of the order of processing and the names of the most commonly functions and their commonly used arguments.

Load data from web services:

```
library(EGRET)
Daily <- readNWISDaily("06934500","00060","1979-10-01","2010-09-30")
Sample <-readNWISSample("06934500","00631","1970-10-01","2011-09-30")
INFO <-readNWISInfo("06934500","00631")
eList <-mergeReport(INFO, Daily, Sample)
```

This is a sample workflow for using WRTDS on the Choptank River at Greensboro, MD, for nitrate:

```
library(EGRET)
############################
# Gather discharge data:
siteID <- "01491000" #Choptank River at Greensboro, MD
startDate <- "" #Gets earliest date
endDate <- "2011-09-30"
# Gather sample data:
parameter_cd<-"00631" #5 digit USGS code
Sample <- readNWISSample(siteID,parameter_cd,startDate,endDate)
#Gets earliest date from Sample record:
#This is just one of many ways to assure the Daily record
#spans the Sample record
startDate <- min(as.character(Sample$Date))
# Gather discharge data:
Daily <- readNWISDaily(siteID,"00060",startDate,endDate)
# Gather site and parameter information:

# Here user must input some values for
# the default (interactive=TRUE)
INFO<- readNWISInfo(siteID,parameter_cd)
# Merge discharge with sample data:
eList <- mergeReport(INFO, Daily, Sample)
############################
############################
# Check sample data:
multiPlotDataOverview(eList)
############################
############################
# Run WRTDS model:
eList <- modelEstimation(eList)
############################
############################
#Check model results:
plotConcTimeDaily(eList)
plotFluxTimeDaily(eList)
plotConcPred(eList)
plotFluxPred(eList)
plotResidPred(eList)
plotResidQ(eList)
plotResidTime(eList)
boxResidMonth(eList)
boxConcThree(eList)
plotConcHist(eList)
plotFluxHist(eList)
```

```
      # Multi-line plots:
      date1 <- "2000-09-01"
      date2 <- "2005-09-01"
      date3 <- "2009-09-01"
      qBottom<-100
      qTop<-5000
      plotConcQSmooth(eList, date1, date2, date3, qBottom, qTop,concMax=2,qUnit=1)
      q1 <- 10
      q2 <- 25
      q3 <- 75
      centerDate <- "07-01"
      yearEnd <- 2009
      yearStart <- 2000
      plotConcTimeSmooth(eList, q1, q2, q3, centerDate, yearStart, yearEnd)
      # Multi-plots:
      fluxBiasMulti(eList)
      #Contour plots:
      clevel<-seq(0,2,0.5)
      maxDiff<-0.8
      yearStart <- 2000
      yearEnd <- 2010
      plotContours(eList, yearStart,yearEnd,qBottom,qTop, contourLevels =
clevel,qUnit=1)
      plotDiffContours(eList, yearStart,yearEnd, qBottom,qTop,maxDiff,qUnit=1) # mod-
ify this for your own computer file structure
      savePath<-"/Users/rhirsch/Desktop/"
      saveResults(savePath, eList)
```

This is a sample workflow for a Flow History application for the entire record.

```
          library(EGRET)
      # Flow history analysis
      ############################
      # Gather discharge data:
      siteID <- "01491000"   #Choptank River at Greensboro, MD
      startDate <- ""
      # Get earliest date
      endDate <- ""   # Get latest date
      Daily <- readNWISDaily(siteID,"00060",startDate,endDate)
      # Gather site and parameter information:
      # Here user must input some values for
      # the default (interactive=TRUE)
      INFO<- readNWISInfo(siteID,”00060”)
      INFO$shortName <- "Choptank River at Greensboro, MD"
      eList <- as.egret(INFO, Daily, NA, NA)
      ############################
      ############################
      # Check flow history data:
      plotFlowSingle(eList, istat=7,qUnit="thousandCfs")
      plotSDLogQ(eList)
      plotQTimeDaily(eList, qLower=1,qUnit=3)
      plotFour(eList qUnit=3)
      plotFourStats(eList, qUnit=3)
      ############################
      # modify this for your own computer file structure:
      savePath<-"/Users/rhirsch/Desktop/"
      saveResults(savePath, eList)
```

Index of Function Names

ISSN 2328-7055 (online)
http://dx.doi.org/10.3133/tm4A10